Manual de PHP

Mehdi Achour
Friedhelm Betz
Antony Dovgal
Nuno Lopes
Philip Olson
Georg Richter
Damien Seguy
Jakub Vrana
Y otros muchos

Editado por

Gabor Hojtsy
Rafael Martínez
Angela Pardo
Federico Finos
Pablo Daniel Rigazzi
Robert Sánchez
Leonardo Boshell
Javier Eguiluz Perez
Javier Tacón Iglesias
Enrique Garcia Briones

21-02-2005

Tabla de contenidos

II. Advanced PHP debugger
III. Funciones de matrices
IV. Funciones Aspell [deprecated]
V. Funciones matemáticas de precisión arbitraria BCMath
VI. PHP bytecode Compiler
VII. Funciones de compresión Bzip2
VIII. Funciones de calendario
IX. Funciones del API de CCVS
X. Classkit Functions
XI. Funciones de Clases/Objetos
XII. Funciones COM y .Net (Windows)
XIII. Funciones ClibPDF
XIV. Crack Functions
XV. Funciones de Tipo de Caracter
XVI. Funciones CURL (Client URL Library)
XVII. Funciones de pago electrónico
XVIII. Cyrus IMAP administration Functions
XIX. Funciones de Fecha y Hora
XX. Funciones de la capa de abstraccion de bases de datos (dbm-style)
XXI. Funciones para dBase
XXII. Funciones DBM Functions [obsoletas]
XXIII. DB++ Functions
XXIV. dbx Functions
XXV. Funciones de acceso directo a E/S
XXVI. Funciones de Directorio
XXVII. DOM Functions
XXVIII. Funciones DOM XML
XXIX. .NET Functions
XXX. Funciones de Gestión de Errores y Registros
XXXI. Funciones de Ejecución de Programas
XXXII. Exif Functions
XXXIII. File Alteration Monitor Functions
XXXIV. FrontBase Functions
XXXV. Funciones del Formato de Datos de Formulario
XXXVI. Funciones filePro
XXXVII. Funciones del Sistema de Archivos
XXXVIII. FriBiDi Functions
XXXIX. Funciones FTP
XL. Funciones de Gestión de Funciones
XLI. Gettext
XLII. GMP Functions
XLIII. Funciones HTTP
XLIV. Funciones para Hyperwave
XLV. Hyperwave API Functions
XLVI. Funciones InterBase
XLVII. Funciones ICAP [obsoletas]
XLVIII. Funciones iconv
XLIX. ID3 Functions
L. Funciones de Informix
LI. IIS Administration Functions
LII. Funciones para imágenes
LIII. Funciones IMAP, POP3 y NNTP
LIV. Opciones e Información de PHP
LV. Ingres II functions
LVI. IRC Gateway Functions

Prefacio

> PHP, acrónimo de "PHP: Hypertext Preprocessor", es un lenguaje "Open Source" interpretado de alto nivel, especialmente pensado para desarrollos web y el cual puede ser embebido en páginas HTML. La mayoría de su sintaxis es similar a C, Java y Perl y es fácil de aprender. La meta de este lenguaje es permitir escribir a los creadores de páginas web, páginas dinámicas de una manera rápida y fácil, aunque se pueda hacer mucho más con PHP.

Este manual contiene principalmente una referencia de funciones PHP, tambien contiene una referencia del lenguaje, explicaciones de caracteristicas importantes de PHP y alguna información suplementaria.

Este manual se puede conseguir en diferentes formatos en http://www.php.net/docs.php. Más información sobre como este manual es desarrollado puede encontrarse en el apéndice 'Sobre este manual'. Si estais interesados en la Historia de PHP, visitar el capítulo correspondiente.

Autores y colaboradores

Actualmente presentamos a los colaboradores mas activos en la portada del manual. Esto no quiere decir que no existan mas colaboradores que tambien trabajan en el manual o que hayan ayudado de manera activa en un pasado. Existen tambien muchas personas que ayudan con sus notas de usuarios a los cuales le estamos infinitamente agradecidos. Todas las listas estan en orden alfabetico.

Autores y editores

Los siguientes colaboradores han ayudado ó ayudan de manera muy activa en la creación de contenidos para el manual: Jouni Ahto, Alexander Aulbach, Daniel Beckham, Stig Bakken, Jesus M. Castagnetto, Ron Chmara, John Coggeshall, Simone Cortesi, Markus Fischer, Wez Furlong, Sara Golemon, Rui Hirokawa, Brad House, Moriyoshi Koizumi, Rasmus Lerdorf, Andrew Lindeman, Stanislav Malyshev, Rafael Martinez, Yasuo Ohgaki, Derick Rethans, Sander Roobol, Egon Schmid, Thomas Schoefbeck, Sascha Schumann, Lars Torben Wilson, Jim Winstead, Jeroen van Wolffelaar, y Andrei Zmievski.

Los siguientes colaboradores han ayudado de manera muy activa en la elaboración del manual: Stig Bakken, Hartmut Holzgraefe, y Egon Schmid.

Encargados de mantenimiento de las Notas de usuarios

Encargados de mantenimiento más activos: Mehdi Achour, Friedhelm Betz, Vincent Gevers, Aidan Lister, Nuno Lopes, y Tom Sommer.

Estas persones han ayudado mucho en el pasaso en el mantenimiento de las Notas de usuarios: Daniel Beckham, Victor Boivie, Jesus M. Castagnetto, Nicolas Chaillan, Ron Chmara, James Cox, Sara Golemon, Zak Greant, Szabolcs Heilig, Oliver Hinckel, Hartmut Holzgraefe, Rasmus Lerdorf, Andrew Lindeman, Maxim Maletsky, James Moore, Sebastian Picklum, Derick Rethans, Sander Roobol, Damien Seguy, Jason Sheets, Jani Taskinen, Yasuo Ohgaki, Philip Olson, Lars Torben Wilson, Jim Winstead, Jared Wyles, y Jeroen van Wolffelaar.

I. Conceptos básicos

Tabla de contenidos

Capítulo 1. Introducción

¿Qué es PHP?

PHP (acrónimo de "PHP: Hypertext Preprocessor") es un lenguaje de "código abierto" interpretado, de alto nivel, embebido en páginas HTML y ejecutado en el servidor.

Una respuesta corta y concisa, pero, ¿qué significa realmente? Un ejemplo nos aclarará las cosas:

Ejemplo 1-1. Un ejemplo introductorio

```
<html>
    <head>
        <title>Ejemplo</title>
    </head>
    <body>

        <?php
        echo "Hola, &iexcl;soy un script PHP!";
        ?>

    </body>
</html>
```

Puede apreciarse que no es lo mismo que un script escrito en otro lenguaje de programación como Perl o C -- En vez de escribir un programa con muchos comandos para crear una salida en HTML, escribimos el código HTML con cierto código PHP embebido (incluido) en el mismo, que producirá cierta salida (en nuestro ejemplo, producirá un texto). El código PHP se incluye entre etiquetas especiales de comienzo y final que nos permitirán entrar y salir del modo PHP.

Lo que distingue a PHP de la tecnología Javascript, la cual se ejecuta en la máquina cliente, es que el código PHP es ejecutado en el servidor. Si tuviésemos un script similar al de nuestro ejemplo en nuestro servidor, el cliente solamente recibiría el resultado de su ejecución en el servidor, sin ninguna posibilidad de determinar qué código ha producido el resultado recibido. El servidor web puede ser incluso configurado para que procese todos los archivos HTML con PHP.

Lo mejor de usar PHP es que es extremadamente simple para el principiante, pero a su vez, ofrece muchas características avanzadas para los programadores profesionales. No sienta miedo de leer la larga lista de características de PHP, en poco tiempo podrá empezar a escribir sus primeros scripts.

Aunque el desarrollo de PHP está concentrado en la programación de scripts en el lado del servidor, se puede utilizar para muchas otras cosas. Siga leyendo y descubra más sobre PHP en la sección ¿Qué se puede hacer con PHP?.

¿Qué se puede hacer con PHP?

PHP puede hacer cualquier cosa que se pueda hacer con un script CGI, como procesar la información de formularios, generar páginas con contenidos dinámicos, o enviar y recibir cookies. Y esto no es todo, se puede hacer mucho más.

Existen tres campos en los que se usan scripts escritos en PHP.

- Scripts del lado del servidor. Este es el campo más tradicional y el principal foco de trabajo. Se necesitan tres cosas para que esto funcione. El intérprete PHP (CGI ó módulo), un servidor web y un navegador. Es necesario correr el servidor web con PHP instalado. El resultado del programa PHP se puede obtener a través del navegador, conectándose con el servidor web. Consultar la sección Instrucciones de instalación para más información.

- Scripts en la línea de comandos. Puede crear un script PHP y correrlo sin ningún servidor web o navegador. Solamente necesita el intérprete PHP para usarlo de esta manera. Este tipo de uso es ideal para scripts ejecutados regularmente desde cron (en *nix o Linux) o el Planificador de tareas (en Windows). Estos scripts también pueden ser usados para tareas simples de procesamiento de texto. Consultar la sección Usos de PHP en la línea de

comandos para más información.

- Escribir aplicaciones de interfaz gráfica. Probablemente PHP no sea el lenguaje más apropiado para escribir aplicaciones gráficas, pero si conoce bien PHP, y quisiera utilizar algunas características avanzadas en programas clientes, puede utilizar PHP-GTK para escribir dichos programas. También es posible escribir aplicaciones independientes de una plataforma. PHP-GTK es una extensión de PHP, no disponible en la distribución principal. Si está interesado en PHP-GTK, puedes visitar las páginas web del proyecto.

PHP puede ser utilizado en cualquiera de los principales sistemas operativos del mercado, incluyendo Linux, muchas variantes Unix (incluyendo HP-UX, Solaris y OpenBSD), Microsoft Windows, Mac OS X, RISC OS y probablemente alguno más. PHP soporta la mayoría de servidores web de hoy en día, incluyendo Apache, Microsoft Internet Information Server, Personal Web Server, Netscape e iPlanet, Oreilly Website Pro server, Caudium, Xitami, OmniHTTPd y muchos otros. PHP tiene módulos disponibles para la mayoría de los servidores, para aquellos otros que soporten el estándar CGI, PHP puede usarse como procesador CGI.

De modo que, con PHP tiene la libertad de elegir el sistema operativo y el servidor de su gusto. También tiene la posibilidad de usar programación procedimental o programación orientada a objetos. Aunque no todas las características estándar de la programación orientada a objetos están implementadas en la versión actual de PHP, muchas bibliotecas y aplicaciones grandes (incluyendo la biblioteca PEAR) están escritas íntegramente usando programación orientada a objetos.

Con PHP no se encuentra limitado a resultados en HTML. Entre las habilidades de PHP se incluyen: creación de imágenes, archivos PDF y películas Flash (usando libswf y Ming) sobre la marcha. Tambié puede presentar otros resultados, como XHTM y archivos XML. PHP puede autogenerar éstos archivos y almacenarlos en el sistema de archivos en vez de presentarlos en la pantalla.

Quizás la característica más potente y destacable de PHP es su soporte para una gran cantidad de bases de datos. Escribir un interfaz vía web para una base de datos es una tarea simple con PHP. Las siguientes bases de datos están soportadas actualmente:

Adabas D	Ingres	Oracle (OCI7 and OCI8)
dBase	InterBase	Ovrimos
Empress	FrontBase	PostgreSQL
FilePro (read-only)	mSQL	Solid
Hyperwave	Direct MS-SQL	Sybase
IBM DB2	MySQL	Velocis
Informix	ODBC	Unix dbm

También contamos con una extensión DBX de abstracción de base de datos que permite usar de forma transparente cualquier base de datos soportada por la extensión. Adicionalmente, PHP soporta ODBC (el Estándar Abierto de Conexión con Bases de Datos), asi que puede conectarse a cualquier base de datos que soporte tal estándar.

PHP también cuenta con soporte para comunicarse con otros servicios usando protocolos tales como LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (en Windows) y muchos otros. También se pueden crear sockets puros. PHP soporta WDDX para el intercambio de datos entre lenguajes de programación en web. Y hablando de interconexión, PHP puede utilizar objetos Java de forma transparente como objetos PHP Y la extensión de CORBA puede ser utilizada para acceder a objetos remotos.

PHP tiene unas características muy útiles para el procesamiento de texto, desde expresiones

regulares POSIX extendidas o tipo Perl hasta procesadores de documentos XML. Para procesar y acceder a documentos XML, soportamos los estándares SAX y DOM. Puede utilizar la extensión XSLT para transformar documentos XML.

Si usa PHP en el campo del comercio electrónico, encontrará muy útiles las funciones Cybercash, CyberMUT, VeriSign Payflow Pro y CCVS para sus programas de pago.

Para terminar, contamos con muchas otras extensiones muy interesantes, las funciones del motor de búsquedas mnoGoSearch, funciones para pasarelas de IRC, utilidades de compresión (gzip, bz2),, conversión de calendarios, traducción .....

Como puede apreciar, esta página no es suficiente para enumerar todas las características y beneficios que PHP ofrece. Consulte las secciones Instalación de PHP y Referencia de las funciones para una explicación de las extensiones mencionadas aquí.

Capítulo 2. Una explicación sencilla

A continuación, le introduciremos a PHP en un pequeño y sencillo manual. Este documento explica cómo crear páginas web dinámicas para Internet con PHP, aunque PHP no solamente está diseñado para la creación de éstas. Consulte la sección titulada ¿Qué se puede hacer con PHP? para más información.

Las páginas web que utilizan PHP son tratadas como páginas de HTML comunes y corrientes, y puede crearlas y editarlas de la misma manera que lo hace con documentos normales de HTML.

¿Qué necesito?

En este manual vamos a asumir que usted cuenta con un servidor que soporta PHP y que todos los archivos con la extensión `.php` son manejados por PHP. En la mayoría de servidores, ésta es la extensión que toman los archivos PHP por defecto, pero pregunte al administrador de su servidor para estar seguro. Si su servidor soporta PHP, entonces no necesita hacer nada, solamente crear sus archivos `.php` y guardarlos en su directorio web, y el servidor, como por arte de magia, los analizará para usted. No hay necesidad de compilar nada, tampoco tiene necesidad de instalar otras herramientas. Mírelo de esta manera, estos archivos de PHP son tan simples como archivos de HTML con una nueva familia de etiquetas que le permiten una gran cantidad de cosas. La mayoría de las compañías de hospedaje de páginas web ofrecen el soporte que necesita para usar PHP, pero si por alguna razón ellos no lo hacen, considere leer la sección titulada Recursos PHP para mas información acerca de compañías de hospedaje que soportan PHP

Digamos que usted tiene limitado acceso a internet y se encuentra programando localmente. En este caso, querrá instalar un servidor de web como Apache, y PHP. Lo más seguro es que también quiera instalar una base de datos como MySQL. Puede instalar estos productos individualmente o simplemente localizar un paquete pre-configurado que automáticamente instale todos estos productos con solamente unos movimientos de su ratón. Es muy fácil instalar un servidor web con soporte para PHP en cualquier sistemas operativo, incluyendo Linux y Windows. En Linux, rpmfind y PBone le ayudarán a encontrar un RPM.

Su primera página con PHP

Comienze por crear un archivo llamado `hola.php` y colocarle en el "directorio raíz" (*DOCUMENT_ROOT*) con el siguiente contenido:

Ejemplo 2-1. Nuestro primer script PHP: `hola.php`

```
<html>
 <head>
  <title>Ejemplo de PHP</title>
 </head>
 <body>
 <?php echo "<p>Hola Mundo</p>"; ?>
 </body>
</html>
```

Utilice su navegador web para acceder al archivo, con la URL terminando en "/hola.php". Si está programando localmente este URL lucirá algo como *http://localhost/hola.php* o *http://127.0.0.1/hola.php* pero esto depende de la configuración de su servidor web. Aunque este tema está fuera del alcance de este tutorial, también puede ver las directivas *DocumentRoot* y *ServerName* en la configuración de su servidor (en Apache, esto es `httpd.conf`). Si todo está configurado correctamente, el archivo será analizado por PHP y el siguiente contenido aparecerá en su navegador:

```
<html>
 <head>
  <title>Ejemplo de PHP</title>
 </head>
 <body>
 <p>Hola Mundo</p>
 </body>
</html>
```

Note que esto no es como los scripts de CGI. El archivo no necesita ninguna clase especial de permisos para ser ejecutado. Piense en ellos como si fueran archivos HTML con un conjunto muy especial de etiquetas disponibles, y que hacen muchas cosas interesantes.

Este programa es extremadamente simple, y no necesita usar PHP para crear una página como ésta. Todo lo que hace es mostrar: *Hola Mundo* usando la sentencia **echo()**.

Si ha intentado usar este ejemplo, y no produjo ningún resultado, preguntando si deseaba descargar el archivo, o mostró todo el archivo como texto, lo más seguro es que PHP no se encuentra habilitado en su servidor. Pídale a su administrador que active esta función por usted, o use el capítulo titulado Instalación en el manual. Si está trabajando localmente, lea también el capítulo dedicado a la instalación, y asegúrese de que todo esté configurado apropiadamente. Si el problema continúa, por favor use una de las muchas opciones para obtener ayuda con PHP.

El objetivo de este ejemplo es demostrar cómo puede usar las etiquetas PHP. En este ejemplo usamos *<?php* para indicar el inicio de la etiqueta PHP. Después indicamos la sentencia y abandonamos el modo PHP usando *?>*. Puede salir de PHP y regresar cuantas veces lo desee usando este método. Para más información, puede leer la sección en el manual titulada Sintaxis básica de PHP.

> **Una nota acerca de editores de texto:** Hay muchos editores de texto y Entornos Integrados de Desarrollo (IDE por sus siglas en Inglés) que puede usar para crear, editar, y organizar archivos PHP. Puede encontrar una lista parcial de éstos en Lista de editores de PHP. Si desea recomendar un editor, por favor visite la página mencionada anteriormente, y comunique su recomendación a las personas encargadas del mantenimiento para que lo incluyan en la lista. Contar con un editor que resalte la sintaxis de PHP puede ser de mucha ayuda.

> **Una nota acerca de los procesadores de palabras:** Los procesadores de palabras como "StarOffice", "Microsoft word" y "Abiword" no son buenas opciones para editar archivos de PHP. Si desea usar uno de éstos programas para probar sus scripts, primero debe asegurarse de guardar el documento en formato de "Texto" puro, o PHP no será capaz de ejecutar el script.
>
> **Una nota acerca del "Bloc de Notas de Windows":** Si desea escribir sus archivos PHP usando el "Bloc de Notas de Windows" o en algún otro editor de texto para Windows necesita asegurarse de que sus archivos sean guardados con la extensión .php (la mayoría de editores de texto en Windows automáticamente tratarán de añadir la extensión .txt a los archivos a menos que tome los siguientes pasos para prevenirlo). Cuando guarde sus archivos y el programa le pregunte qué nombre le desea dar al archivo, use comillas para indicar el nombre (es decir, "`hola.php`"). Una alternativa es, en la lista de opciones "Archivos de Texto *.txt", seleccionar la opción "Todos los archivos *.*". Aquí puede escribir el nombre del archivo sin las comillas.

Ahora que ha creado un pequeño script de PHP que funciona correctamente, es hora de trabajar con el script de PHP más famoso; vamos a hacer una llamada a la función **phpinfo()** para obtener información acerca de su sistema y configuración como las variables predefinidas disponibles, los módulos utilizados por PHP, y las diferentes opciones de configuración. Tomemos unos segundos para revisar esta información.

Algo útil

Hagamos ahora algo que puede ser más útil. Vamos a chequear qué clase de navegador web utiliza. Para hacerlo, vamos a consultar la información que el navegador nos envía como parte de su petición HTTP. Esta información es guardada en una variable. Las variables siempre comienzan con un signo de dólar ("$") en PHP. La variable que vamos a utilizar en esta situación es *$_SERVER ["HTTP_USER_AGENT"]*.

> **Nota:** $_SERVER es una variable reservada por PHP que contiene toda la información del servidor web. Es conocida como Autoglobal (o Superglobal). Consulte el manual en su sección titulada Autoglobales para mas información. Éstas son variables especiales que fueron introducidas en la versión 4.1.0 de PHP. Antes podíamos usar las matrices *$HTTP_*_VARS*, tales como *$HTTP_SERVER_VARS*. Aunque éstas han sido marcadas como obsoletas, tales matrices todavía existen. (También puede echar un vistazo a las notas relacionadas con el código antiguo.)

Para poder ver esta variable solo necesita:

Ejemplo 2-2. Impresión de una variable (elemento de la matriz)

```
<?php echo $_SERVER["HTTP_USER_AGENT"]; ?>
```

Un ejemplo de los resultados de este programa sería:

```
Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
```

Hay muchos tipos de variables en PHP. En el ejemplo anterior imprimimos una matriz. Las matrices pueden ser muy útiles.

$_SERVER es simplemente una variable que se encuentra disponible automáticamente para usted en PHP. Puede encontrar una lista en la sección titulada Variables Reservadas del manual, o puede generar una lista completa creando un archivo como el presentado a continuación:

Ejemplo 2-3. Consultar todas las variables predefinidas con phpinfo()

```
<?php phpinfo(); ?>
```

Si abre este archivo con su navegador, puede ver una página con información acerca de PHP, junto a una lista de todas las variables que puede usar.

Puede usar más de un declaración PHP dentro de una etiqueta PHP, y crear pequeños segmentos de código que pueden hacer más que un "echo". Por ejemplo, si quisiéramos detectar el uso de "Internet Explorer", haríamos algo así:

Ejemplo 2-4. Ejemplos de uso de estructuras de control y funciones

```
<?php
if (strstr($_SERVER["HTTP_USER_AGENT"], "MSIE")) {
        echo "Est&aacute; usando Internet Explorer<br />";
}
?>
```

Un ejemplo de los resultado del script puede ser:

```
Est&aacute; usando Internet Explorer<br />
```

A continuación introduciremos un par de conceptos nuevos. Tenemos una declaración "if". Si está familiarizado con la sintaxis básica del lenguaje "C", esto se vera lógico, Pero si no entiende "C", u otros lenguajes de programación donde encuentra la sintaxis usada anteriormente, probablemente debería conseguir un libro que le introduzca mejor a PHP, y lea los primeros capítulos, o también puede ver la parte del manual titulada Referencia del lenguaje. Puedes encontrar una lista de libros sobre PHP en http://www.php.net/books.php.

El segundo concepto que introducimos fue la función llamada **strstr()**. **strstr()** es una función integrada de PHP que busca un cadena dentro de otra cadena más larga. En el caso anterior estamos buscando *"MSIE"* dentro de *$_SERVER["HTTP_USER_AGENT"]*. Si la cadena fue encontrada, la función devolverá verdadero ("**TRUE**"), la declaración "if" se evalúa a verdadero ("**TRUE**") y el código adentro de las llaves {} es ejecutado. De otra manera no resulta ejecutado. Tómese la libertad de crear ejemplos similares usando "if", "else" ("de otra manera"), y otras funciones como **strtoupper()** y **strlen()**. Cada página del manual contiene ejemplos que puede usar. Si no está seguro sobre el modo de uso éstas funciones, es recomendable que lea las páginas del manual tituladas Cómo leer una definición de función y la sección relacionada a Funciones en PHP

Podemos continuar y demostrar cómo puede saltar adentro y afuera del modo PHP en el medio de un bloque de código.

Ejemplo 2-5. Mezcla de los modos HTML y PHP

```
<?php
if (strstr($_SERVER["HTTP_USER_AGENT"], "MSIE")) {
?>
<h3>strstr debe haber devuelto verdadero</h3>
<center><b>Est&aacute; usando Internet Explorer</b></center>
<?php
} else {
?>
<h3>strstr debi&oacute; devolver falso</h3>
<center><b>No est&aacute; usando Internet Explorer</b></center>
<?php
}
?>
```

Un ejemplo de los resultados de este script puede ser:

```
<h3>strstr debe haber devuelto verdadero </h3>
<center><b>Est&aacute; usando Internet Explorer</b></center>
```

En vez de usar una sentencia PHP "echo" para demostrar algo, saltamos fuera del código PHP y escribimos HTML puro. Este es un punto muy importante y potente que debemos observar aquí, y

es que la fluidez lógica del script está intacta. Sólamente las partes donde hay HTML serán enviadas a su navegador dependiendo de los resultados que **strstr()** devuelve (si fue verdadero [**TRUE**], o falso [**FALSE**]). En otras palabras, si la cadena *MSIE* fue encontrada o no.

Uso de Formularios HTML

Otra de las características de PHP es que gestiona formularios de HTML. El concepto básico que es importante entender es que cualquier elemento de los formularios estará disponible automáticamente en su código PHP. Por favor refiérase a la sección titulada Variables fuera de PHP en el manual para más información y ejemplos sobre cómo usar formularios HTML con PHP. Observemos un ejemplo:

Ejemplo 2-6. Un formulario HTML sencillo

```
<form action="accion.php" method="POST">
 Su nombre: <input type="text" name="nombre" />
 Su edad: <input type="text" name="edad" />
 <input type="submit">
</form>
```

No hay nada especial en este formularo, es HTML limpio sin ninguna clase de etiquetas desconocidas. Cuando el cliente llena éste formulario y oprime el botón etiquetado "Submit", una página titulada `accion.php` es llamada. En este archivo encontrará algo así:

Ejemplo 2-7. Procesamiento de información de nuestro formulario HTML

```
Hola <?php echo $_POST["nombre"]; ?>.
Tiene <?php echo $_POST["edad"]; ?> a&ntilde;os
```

Un ejemplo del resultado de este script podría ser:

```
Hola José.
Tiene 22 años
```

Es aparentemente obvio lo que hace. No hay mucho más que decir al respecto. Las variables *$_POST["nombre"]* y *$_POST["edad"]* son definidas automáticamente por PHP. Hace un momento usamos la variable autoglobal *$_SERVER*, ahora hemos introducido autoglobal $_POST, que contiene toda la información enviada por el método POST. Fíjese en el atributo *method* en nuestro formulario; es POST. Si hubiéramos usado *GET*, entonces nuestra información estaría en la variable autoglobal $_GET. También puede utilizar la autoglobal $_REQUEST si no le importa el origen de la petición. Ésta variable contiene una mezcla de información GET, POST y COOKIE. También puede ver la función **import_request_variables()**.

Use de código antiguo con nuevas versiones de PHP

Ahora que PHP ha crecido y se ha convertido en un lenguaje popular, hay muchos más recursos que contienen código que puede reusar en sus propios programas. Por lo general, las personas que se encargan del desarrollo de PHP tratan de que el lenguaje sea compatible con versiones anteriores, para que los programas escritos con versiones antiguas continúen funcionando cuando instale una nueva versión de PHP. En un mundo perfecto, nunca necesitaría modificar su código para hacerlo funcionar con versiones nuevas del lenguaje; pero, como todos sabemos, este no es un mundo perfecto, y usualmente son necesarios los cambios en su código.

Dos de los cambios mas importantes que afectan el código viejo son:

- La desaparición de las matrices *$HTTP_*_VARS* (que usualmente son usadas como globales

al interior de una función o método). Las siguientes matrices autoglobales fueron introducidas en la versión 4.1.0 de PHP. Estas son: *$_GET*, *$_POST*, *$_COOKIE*, *$_SERVER*, *$_FILES*, *$_ENV*, *$_REQUEST*, y *$_SESSION*. Las antiguas *$HTTP_*_VARS*, como $HTTP_POST_VARS, todavía existen, y han existido desde PHP 3.

- Las variables externas que ya no son registradas automáticamente. En otras palabras, a partir de PHP 4.2.0, la directiva PHP register_globals está *deshabilitada* (su valor es off) en `php.ini`. El método preferido para obtener acceso a éstos valores es por medio de las "matrices autoglobales" mencionados anteriormente. Los scripts, libros y tutoriales antiguos pueden asumir que ésta directiva es definida automáticamente como "on". Si es así, puede usar, por ejemplo, *$id* desde la URL *http://www.example.com/foo.php?id=42*. Por otra parte, no importa si el valor de la directiva es "on" u "off", *$_GET['id']* está siempre disponible.

Para más detalles relacionados con estos cambios, puede ver la sección sobre variables predefinidas.

¿Y ahora qué?

Con lo que hemos visto, puede entender la mayor parte del manual, y también los ejemplos que están disponibles en los archivos. También puede encontrar otros ejemplos en los diferentes sitios de php.net en la sección de enlaces: http://www.php.net/links.php.

Para ver una presentación que muestra más acerca de lo que puede hacer PHP, visite los diferentes sitios con material relacionado a las conferencias realizadas: http://conf.php.net/ y http://talks.php.net/.

II. Instalación y configuración

Tabla de contenidos

Capítulo 3. Consideraciones generales de instalación

Antes de instalar PHP, necesitais saber porque quereis utilizarlo. Existen tres campos principales en donde PHP se puede usar, tal y como se describe en la sección Qué se puede hacer con PHP?:

- Scripts en la parte del servidor
- Scripts en linea de comandos
- Aplicaciones gráficas clientes

El primero es el más tradicional y el principal campo de trabajo. Se necesitan tres cosas para que funcione. El analizador PHP (CGI ó módulo), un servidor web y un navegador. Dependiendo de la versión de sistema operativo que utiliceis, probablemente tengais un servidor web (p.ej: Apache en Linux y MacOS X ó IIS en Windows). También se puede alquilar espacio web en una empresa que ofrezca este servicio. De esta manera no se necesita instalar nada, solamente escribir los scripts PHP, subirlos al espacio alquilado y ver el resultado en vuestro navegador.

Teneis dos maneras de utilizar PHP, si instalais vosotros el servidor y PHP. Existen módulos directos (también llamados SAPI) para muchos servidores web, como Apache, Microsoft Internet Information Server, Netscape e iPlanet. Muchos otros servidores soportan ISAPI, (p.ej.:OmniHTTPd). Si PHP no soporta un módulo para tu servidor web, siempre se puede usar como binario CGI. Esto significa que el servidor se configura para usar el ejecutable para linea de comandos de PHP en el procesado de peticiones de ficheros PHP.

Si estais interesados en usar PHP desde la linea de comandos (p.ej.: para generar imagenes offline ó procesar ficheros de textos, etc) necesitais el ejecutable para linea de comandos. Para más información, lea la sección Usando PHP desde la línea de comandos. En este caso no se necesita ni servidor web, ni navegador.

Con PHP tambien se puede escribir aplicaciones gráficas usando la extensión PHP-GTK. Esta es una forma totalmente distinta de utilizar PHP que cuando se utiliza para escribir páginas web, ya que no se genera código HTML sino que se trabaja con ventanas y objetos dentro de las mismas. Para más información sobre PHP-GTK, visitar las páginas dedicadas a esta extensión. PHP-GTK no se incluye en la distribución oficial de PHP.

A partir de ahora, esta sección tratará sobre la configuración de PHP con servidores web, en Unix y Windows, tanto como módulo, como binario CGI.

PHP, el código fuente y distribuciones binarias para Windows se pueden encontrar en http://www.php.net/. Recomendamos utilizar un servidor espejo cerca de donde esteis para bajaros la versión de PHP que querais.

Capítulo 4. Installation on Unix systems

This section will guide you through the general configuration and installation of PHP on Unix systems. Be sure to investigate any sections specific to your platform or web server before you begin the process.

As our manual outlines in the General Installation Considerations section, we are mainly dealing with web centric setups of PHP in this section, although we will cover setting up PHP for command line usage as well.

There are several ways to install PHP for the Unix platform, either with a compile and configure process, or through various pre-packaged methods. This documentation is mainly focused around the process of compiling and configuring PHP. Many Unix like systems have some sort of package installation system. This can assist in setting up a standard configuration, but if you need to have a different set of features (such as a secure server, or a different database driver), you may need to build PHP and/or your webserver. If you are unfamiliar with building and compiling your own software, it is worth checking to see whether somebody has already built a packaged version of PHP with the features you need.

Prerequisite knowledge and software for compiling:

- Basic Unix skills (being able to operate "make" and a C compiler)
- An ANSI C compiler
- flex: Version 2.5.4
- bison: Version 1.28 (preferred), 1.35, or 1.75
- A web server
- Any module specific components (such as gd, pdf libs, etc.)

The initial PHP setup and configuration process is controlled by the use of the commandline options of the **configure** script. You could get a list of all available options along with short explanations running **./configure --help**. Our manual documents the different options separately. You will find the core options in the appendix, while the different extension specific options are descibed on the reference pages.

When PHP is configured, you are ready to build the module and/or executables. The command **make** should take care of this. If it fails and you can't figure out why, see the Problems section.

Apache 1.3.x on Unix systems

This section contains notes and hints specific to Apache installs of PHP on Unix platforms. We also have instructions and notes for Apache 2 on a separate page.

You can select arguments to add to the **configure** on line 10 below from the list of core configure options and from extension specific options described at the respective places in the manual. The version numbers have been omitted here, to ensure the instructions are not incorrect. You will need to replace the 'xxx' here with the correct values from your files.

Ejemplo 4-1. Installation Instructions (Apache Shared Module Version) for PHP

1. gunzip apache_xxx.tar.gz
2. tar -xvf apache_xxx.tar
3. gunzip php-xxx.tar.gz
4. tar -xvf php-xxx.tar
5. cd apache_xxx
6. ./configure --prefix=/www --enable-module=so
7. make
8. make install
9. cd ../php-xxx

10. Now, configure your PHP. This is where you customize your PHP with various options, like which extensions will be enabled. Do a ./configure --help for a list of available options. In our example we'll do a simple configure with Apache 1 and MySQL support. Your path to apxs may differ from our example.

```
./configure --with-mysql --with-apxs=/www/bin/apxs
```

11. make
12. make install

    If you decide to change your configure options after installation, you only need to repeat the last three steps. You only need to restart apache for the new module to take effect. A recompile of Apache is not needed.

    Note that unless told otherwise, 'make install' will also install PEAR, various PHP tools such as phpize, install the PHP CLI, and more.

13. Setup your php.ini file:

```
cp php.ini-dist /usr/local/lib/php.ini
```

    You may edit your .ini file to set PHP options. If you prefer your php.ini in another location, use --with-config-file-path=/some/path in step 10.

    If you instead choose php.ini-recommended, be certain to read the list of changes within, as they affect how PHP behaves.

14. Edit your httpd.conf to load the PHP module. The path on the right hand side of the LoadModule statement must point to the path of the PHP module on your system. The make install from above may have already added this for you, but be sure to check.

    For PHP 4:

```
LoadModule php4_module libexec/libphp4.so
```

    For PHP 5:

```
LoadModule php5_module libexec/libphp5.so
```

15. And in the AddModule section of httpd.conf, somewhere under the ClearModuleList, add this:

    For PHP 4:

```
AddModule mod_php4.c
```

    For PHP 5:

```
AddModule mod_php5.c
```

16. Tell Apache to parse certain extensions as PHP. For example, let's have Apache parse the .php extension as PHP. You could have any extension(s) parse as PHP by simply adding more, with each separated by a space. We'll add .phtml to demonstrate.

```
AddType application/x-httpd-php .php .phtml
```

    It's also common to setup the .phps extension to show highlighted PHP

Alternatively, to install PHP as a static object:

Ejemplo 4-2. Installation Instructions (Static Module Installation for Apache) for PHP

```
1.  gunzip -c apache_1.3.x.tar.gz | tar xf -
2.  cd apache_1.3.x
3.  ./configure
4.  cd ..

5.  gunzip -c php-4.x.y.tar.gz | tar xf -
6.  cd php-4.x.y
7.  ./configure --with-mysql --with-apache=../apache_1.3.x
8.  make
9.  make install

10. cd ../apache_1.3.x

11. ./configure --prefix=/www --activate-module=src/modules/php4/libphp4.a
    (The above line is correct! Yes, we know libphp4.a does not exist at this
    stage. It isn't supposed to. It will be created.)

12. make
    (you should now have an httpd binary which you can copy to your Apache bin dir if
    is is your first install then you need to "make install" as well)

13. cd ../php-4.x.y
14. cp php.ini-dist /usr/local/lib/php.ini

15. You can edit /usr/local/lib/php.ini file to set PHP options.
    Edit your httpd.conf or srm.conf file and add:
    AddType application/x-httpd-php .php
```

Depending on your Apache install and Unix variant, there are many possible ways to stop and restart the server. Below are some typical lines used in restarting the server, for different apache/unix installations. You should replace */path/to/* with the path to these applications on your systems.

Ejemplo 4-3. Example commands for restarting Apache

```
1. Several Linux and SysV variants:
/etc/rc.d/init.d/httpd restart

2. Using apachectl scripts:
/path/to/apachectl stop
/path/to/apachectl start

3. httpdctl and httpsdctl (Using OpenSSL), similar to apachectl:
/path/to/httpsdctl stop
/path/to/httpsdctl start

4. Using mod_ssl, or another SSL server, you may want to manually
stop and start:
/path/to/apachectl stop
/path/to/apachectl startssl
```

The locations of the apachectl and http(s)dctl binaries often vary. If your system has *locate* or *whereis* or *which* commands, these can assist you in finding your server control programs.

Different examples of compiling PHP for apache are as follows:

```
./configure --with-apxs --with-pgsql
```

This will create a `libphp4.so` shared library that is loaded into Apache using a LoadModule line in Apache's `httpd.conf` file. The PostgreSQL support is embedded into this `libphp4.so` library.

```
./configure --with-apxs --with-pgsql=shared
```

This will create a `libphp4.so` shared library for Apache, but it will also create a `pgsql.so` shared library that is loaded into PHP either by using the extension directive in `php.ini` file or by loading it explicitly in a script using the **dl()** function.

```
./configure --with-apache=/path/to/apache_source --with-pgsql
```

This will create a `libmodphp4.a` library, a `mod_php4.c` and some accompanying files and copy this into the *src/modules/php4* directory in the Apache source tree. Then you compile Apache using *--activate-module=src/modules/php4/libphp4.a* and the Apache build system will create `libphp4.a` and link it statically into the `httpd` binary. The PostgreSQL support is included directly into this `httpd` binary, so the final result here is a single `httpd` binary that includes all of Apache and all of PHP.

```
./configure --with-apache=/path/to/apache_source --with-pgsql=shared
```

Same as before, except instead of including PostgreSQL support directly into the final `httpd` you will get a `pgsql.so` shared library that you can load into PHP from either the `php.ini` file or directly using **dl()**.

When choosing to build PHP in different ways, you should consider the advantages and drawbacks of each method. Building as a shared object will mean that you can compile apache separately, and don't have to recompile everything as you add to, or change, PHP. Building PHP into apache (static method) means that PHP will load and run faster. For more information, see the Apache webpage on DSO support.

> **Nota:** Apache's default `httpd.conf` currently ships with a section that looks like this:
>
> ```
> User nobody
> Group "#-1"
> ```
>
> Unless you change that to "Group nogroup" or something like that ("Group daemon" is also very common) PHP will not be able to open files.
>
> **Nota:** Make sure you specify the installed version of apxs when using *--with-apxs=/path/to/apxs*. You must NOT use the apxs version that is in the apache sources but the one that is actually installed on your system.

Apache 2.0 on Unix systems

This section contains notes and hints specific to Apache 2.0 installs of PHP on Unix systems.

Aviso
No usar Apache 2.0 y PHP en un sistema en produccion, tanto en Unix como en Windows. Consultar la FAQ

You are highly encouraged to take a look at the Apache Documentation to get a basic understanding of the Apache 2.0 Server.

> **PHP and Apache 2.0.x compatibility notes:** The following versions of PHP are known to work with the most recent version of Apache 2.0.x:
>
> - PHP 4.3.0 or later available at http://www.php.net/downloads.php.
> - the latest stable development version. Get the source code http://snaps.php.net/php4-latest.tar.gz or download binaries for Windows

http://snaps.php.net/win32/php4-win32-latest.zip.

- a prerelease version downloadable from http://qa.php.net/.
- you have always the option to obtain PHP through anonymous CVS.

These versions of PHP are compatible to Apache 2.0.40 and later.

Apache 2.0 *SAPI*-support started with PHP 4.2.0. PHP 4.2.3 works with Apache 2.0.39, don't use any other version of Apache with PHP 4.2.3. However, the recommended setup is to use PHP 4.3.0 or later with the most recent version of Apache2.

All mentioned versions of PHP will work still with Apache 1.3.x.

Download the most recent version of Apache 2.0 and a fitting PHP version from the above mentioned places. This quick guide covers only the basics to get started with Apache 2.0 and PHP. For more information read the Apache Documentation. The version numbers have been omitted here, to ensure the instructions are not incorrect. You will need to replace the 'NN' here with the correct values from your files.

Ejemplo 4-4. Installation Instructions (Apache 2 Shared Module Version)

1. gzip -d httpd-2_0_NN.tar.gz
2. tar xvf httpd-2_0_NN.tar
3. gunzip php-NN.tar.gz
4. tar -xvf php-NN.tar
5. cd httpd-2_0_NN
6. ./configure --enable-so
7. make
8. make install

    Now you have Apache 2.0.NN available under /usr/local/apache2,
    configured with loadable module support and the standard MPM prefork.
    To test the installation use your normal procedure for starting
    the Apache server, e.g.:
    /usr/local/apache2/bin/apachectl start
    and stop the server to go on with the configuration for PHP:
    /usr/local/apache2/bin/apachectl stop.

9. cd ../php-NN

10. Now, configure your PHP. This is where you customize your PHP
    with various options, like which extensions will be enabled. Do a
    ./configure --help for a list of available options. In our example
    we'll do a simple configure with Apache 2 and MySQL support. Your
    path to apxs may differ, in fact, the binary may even be named apxs2 on
    your system.

      ./configure --with-apxs2=/usr/local/apache2/bin/apxs --with-mysql

11. make
12. make install

    If you decide to change your configure options after installation,
    you only need to repeat the last three steps. You only need to
    restart apache for the new module to take effect. A recompile of
    Apache is not needed.

    Note that unless told otherwise, 'make install' will also install PEAR,
    various PHP tools such as phpize, install the PHP CLI, and more.

13. Setup your php.ini

    cp php.ini-dist /usr/local/lib/php.ini

    You may edit your .ini file to set PHP options. If you prefer having
    php.ini in another location, use --with-config-file-path=/some/path in
    step 10.

    If you instead choose php.ini-recommended, be certain to read the list
    of changes within, as they affect how PHP behaves.

14. Edit your httpd.conf to load the PHP module. The path on the right hand
    side of the LoadModule statement must point to the path of the PHP
    module on your system. The make install from above may have already
    added this for you, but be sure to check.

    For PHP 4:

      LoadModule php4_module libexec/libphp4.so

    For PHP 5:

      LoadModule php5_module libexec/libphp5.so

15. Tell Apache to parse certain extensions as PHP. For example,
    let's have Apache parse the .php extension as PHP. You could
    have any extension(s) parse as PHP by simply adding more, with
    each separated by a space. We'll add .phtml to demonstrate.

      AddType application/x-httpd-php .php .phtml

    It's also common to setup the .phps extension to show highlighted PHP
    source, this can be done with:

Following the steps above you will have a running Apache 2.0 with support for PHP as *SAPI* module. Of course there are many more configuration options available for both, Apache and PHP. For more information use **./configure --help** in the corresponding source tree. In case you wish to build a multithreaded version of Apache 2.0 you must overwrite the standard MPM-Module `prefork` either with `worker` or `perchild`. To do so append to your configure line in step 6 above either the option *--with-mpm=worker* or *--with-mpm=perchild*. Take care about the consequences and understand what you are doing. For more information read the Apache documentation about the MPM-Modules.

> **Nota:** If you want to use content negotiation, read related FAQ.

> **Nota:** To build a multithreaded version of Apache your system must support threads. This also implies to build PHP with experimental Zend Thread Safety (ZTS). Therefore not all extensions might be available. The recommended setup is to build Apache with the standard `prefork` MPM-Module.

Caudium

PHP 4 can be built as a Pike module for the Caudium webserver. Note that this is not supported with PHP 3. Follow the simple instructions below to install PHP 4 for Caudium.

Ejemplo 4-5. Caudium Installation Instructions

```
1.  Make sure you have Caudium installed prior to attempting to
    install PHP 4. For PHP 4 to work correctly, you will need Pike
    7.0.268 or newer. For the sake of this example we assume that
    Caudium is installed in /opt/caudium/server/.
2.  Change directory to php-x.y.z (where x.y.z is the version number).
3.  ./configure --with-caudium=/opt/caudium/server
4.  make
5.  make install
6.  Restart Caudium if it's currently running.
7.  Log into the graphical configuration interface and go to the
    virtual server where you want to add PHP 4 support.
8.  Click Add Module and locate and then add the PHP 4 Script Support module.
9.  If the documentation says that the 'PHP 4 interpreter isn't
    available', make sure that you restarted the server. If you did
    check /opt/caudium/logs/debug/default.1 for any errors related to
    <filename>PHP4.so</filename>. Also make sure that
    <filename>caudium/server/lib/[pike-version]/PHP4.so</filename>
    is present.
10. Configure the PHP Script Support module if needed.
```

You can of course compile your Caudium module with support for the various extensions available in PHP 4. See the reference pages for extension specific configure options.

> **Nota:** When compiling PHP 4 with MySQL support you must make sure that the normal MySQL client code is used. Otherwise there might be conflicts if your Pike already has MySQL support. You do this by specifying a MySQL install directory the *--with-mysql* option.

---

fhttpd related notes

To build PHP as an fhttpd module, answer "yes" to "Build as an fhttpd module?" (the *--with-fhttpd=DIR* option to configure) and specify the fhttpd source base directory. The default directory is `/usr/local/src/fhttpd`. If you are running fhttpd, building PHP as a module will give

better performance, more control and remote execution capability.

> **Nota:** Support for fhttpd is no longer available as of PHP 4.3.0.

Sun, iPlanet and Netscape servers on Sun Solaris

This section contains notes and hints specific to Sun Java System Web Server, Sun ONE Web Server, iPlanet and Netscape server installs of PHP on Sun Solaris.

From PHP 4.3.3 on you can use PHP scripts with the NSAPI module to generate custom directory listings and error pages. Additional functions for Apache compatibility are also available. For support in current webservers read the note about subrequests.

You can find more information about setting up PHP for the Netscape Enterprise Server (NES) here: http://benoit.noss.free.fr/php/install-php4.html

To build PHP with Sun JSWS/Sun ONE WS/iPlanet/Netscape webservers, enter the proper install directory for the --with-nsapi=[DIR] option. The default directory is usually `/opt/netscape/suitespot/`. Please also read `/php-xxx-version/sapi/nsapi/nsapi-readme.txt`.

1. Install the following packages from http://www.sunfreeware.com/ or another download site:

   `autoconf-2.13`
   `automake-1.4`
   `bison-1_25-sol26-sparc-local`
   `flex-2_5_4a-sol26-sparc-local`
   `gcc-2_95_2-sol26-sparc-local`
   `gzip-1.2.4-sol26-sparc-local`
   `m4-1_4-sol26-sparc-local`
   `make-3_76_1-sol26-sparc-local`
   `mysql-3.23.24-beta` (if you want mysql support)
   `perl-5_005_03-sol26-sparc-local`
   `tar-1.13` (GNU tar)
2. Make sure your path includes the proper directories *PATH=.:/usr/local/bin:/usr/sbin:/usr/bin:/usr/ccs/bin* and make it available to your system `export PATH`.

3. `gunzip php-x.x.x.tar.gz` (if you have a .gz dist, otherwise go to 4).

4. `tar xvf php-x.x.x.tar`

5. Change to your extracted PHP directory: `cd ../php-x.x.x`

6. For the following step, make sure `/opt/netscape/suitespot/` is where your netscape server is installed. Otherwise, change to the correct path and run:
   ```
   ./configure --with-mysql=/usr/local/mysql \
   --with-nsapi=/opt/netscape/suitespot/ \
   --enable-libgcc
   ```
7. Run **make** followed by **make install**.

After performing the base install and reading the appropriate readme file, you may need to perform some additional configuration steps.

Configuration Instructions for Sun/iPlanet/Netscape. Firstly you may need to add some paths to the *LD_LIBRARY_PATH* environment for the server to find all the shared libs. This can best done in the start script for your webserver. The start script is often located in: `/path/to/server/https-servername/start`. You may also need to edit the configuration files that are located in: `/path/to/server/https-servername/config/`.

1. Add the following line to `mime.types` (you can do that by the administration server):
   ```
   type=magnus-internal/x-httpd-php exts=php
   ```
2. Edit `magnus.conf` (for servers >= 6) or `obj.conf` (for servers < 6) and add the following, shlib will vary depending on your system, it will be something like `/opt/netscape/suitespot/bin/libphp4.so`. You should place the following lines after *mime types init*.
   ```
   Init fn="load-modules" funcs="php4_init,php4_execute,php4_auth_trans" shlib="/opt/net
   Init fn="php4_init" LateInit="yes" errorString="Failed to initialize PHP!" [php_ini="
   ```
   (PHP >= 4.3.3) The *php_ini* parameter is optional but with it you can place your `php.ini` in your webserver config directory.

3. Configure the default object in `obj.conf` (for virtual server classes [version 6.0+] in their `vserver.obj.conf`):
   ```
   <Object name="default">
   .
   .
   .
   .#NOTE this next line should happen after all 'ObjectType' and before all 'AddLog' li
   Service fn="php4_execute" type="magnus-internal/x-httpd-php" [inikey=value inikey=val
   .
   .
   </Object>
   ```
   (PHP >= 4.3.3) As additional parameters you can add some special `php.ini`-values, for example you can set a *docroot="/path/to/docroot"* specific to the context *php4_execute* is called. For boolean ini-keys please use 0/1 as value, not *"On","Off",...* (this will not work correctly), e.g. *zlib.output_compression=1* instead of *zlib.output_compression="On"*

4. This is only needed if you want to configure a directory that only consists of PHP scripts (same like a `cgi-bin` directory):
   ```
   <Object name="x-httpd-php">
   ObjectType fn="force-type" type="magnus-internal/x-httpd-php"
   Service fn=php4_execute [inikey=value inikey=value ...]
   </Object>
   ```
   After that you can configure a directory in the Administration server and assign it the style *x-httpd-php*. All files in it will get executed as PHP. This is nice to hide PHP usage by renaming files to `.html`.

5. Setup of authentication: PHP authentication cannot be used with any other authentication. ALL AUTHENTICATION IS PASSED TO YOUR PHP SCRIPT. To configure PHP Authentication for the entire server, add the following line to your default object:
   ```
   <Object name="default">
   AuthTrans fn=php4_auth_trans
   .
   .
   .
   </Object>
   ```
6. To use PHP Authentication on a single directory, add the following:

```
<Object ppath="d:\path\to\authenticated\dir\*">
AuthTrans fn=php4_auth_trans
</Object>
```

Nota: The stacksize that PHP uses depends on the configuration of the webserver. If you get crashes with very large PHP scripts, it is recommended to raise it with the Admin Server (in the section "MAGNUS EDITOR").

CGI environment and recommended modifications in php.ini

Important when writing PHP scripts is the fact that Sun JSWS/Sun ONE WS/iPlanet/Netscape is a multithreaded web server. Because of that all requests are running in the same process space (the space of the webserver itself) and this space has only one environment. If you want to get CGI variables like *PATH_INFO*, *HTTP_HOST* etc. it is not the correct way to try this in the old PHP 3.x way with **getenv()** or a similar way (register globals to environment, *$_ENV*). You would only get the environment of the running webserver without any valid CGI variables!

Nota: Why are there (invalid) CGI variables in the environment?

Answer: This is because you started the webserver process from the admin server which runs the startup script of the webserver, you wanted to start, as a CGI script (a CGI script inside of the admin server!). This is why the environment of the started webserver has some CGI environment variables in it. You can test this by starting the webserver not from the administration server. Use the command line as root user and start it manually - you will see there are no CGI-like environment variables.

Simply change your scripts to get CGI variables in the correct way for PHP 4.x by using the superglobal *$_SERVER*. If you have older scripts which use *$HTTP_HOST*, etc., you should turn on *register_globals* in `php.ini` and change the variable order too (important: remove *"E"* from it, because you do not need the environment here):

```
variables_order = "GPCS"
register_globals = On
```

Special use for error pages or self-made directory listings (PHP >= 4.3.3)

You can use PHP to generate the error pages for *"404 Not Found"* or similar. Add the following line to the object in `obj.conf` for every error page you want to overwrite:

```
Error fn="php4_execute" code=XXX script="/path/to/script.php" [inikey=value inikey=valu
```

where *XXX* is the HTTP error code. Please delete any other *Error* directives which could interfere with yours. If you want to place a page for all errors that could exist, leave the *code* parameter out. Your script can get the HTTP status code with *$_SERVER['ERROR_TYPE']*.

Another possibility is to generate self-made directory listings. Just create a PHP script which displays a directory listing and replace the corresponding default Service line for *type="magnus-internal/directory"* in `obj.conf` with the following:

```
Service fn="php4_execute" type="magnus-internal/directory" script="/path/to/script.php"
```

For both error and directory listing pages the original URI and translated URI are in the variables *$_SERVER['PATH_INFO']* and *$_SERVER['PATH_TRANSLATED']*.

Note about nsapi_virtual() and subrequests (PHP >= 4.3.3)

The NSAPI module now supports the **nsapi_virtual()** function (alias: **virtual()**) to make subrequests on the webserver and insert the result in the webpage. This function uses some undocumented features from the NSAPI library. On Unix the module automatically looks for the needed functions and uses them if available. If not, **nsapi_virtual()** is disabled.

> **Nota:** But be warned: Support for **nsapi_virtual()** is EXPERIMENTAL!!!

CGI and commandline setups

The default is to build PHP as a CGI program. This creates a commandline interpreter, which can be used for CGI processing, or for non-web-related PHP scripting. If you are running a web server PHP has module support for, you should generally go for that solution for performance reasons. However, the CGI version enables users to run different PHP-enabled pages under different user-ids.

Aviso
Al usar el modo CGI, vuestro servidor esta expuesto a diferentes ataques. Por favor, leer la sección Seguridad con CGI para aprender como defenderse de estos ataques.

As of PHP 4.3.0, some important additions have happened to PHP. A new SAPI named CLI also exists and it has the same name as the CGI binary. What is installed at *{PREFIX}/bin/php* depends on your configure line and this is described in detail in the manual section named Using PHP from the command line. For further details please read that section of the manual.

Testing

If you have built PHP as a CGI program, you may test your build by typing **make test**. It is always a good idea to test your build. This way you may catch a problem with PHP on your platform early instead of having to struggle with it later.

Benchmarking

If you have built PHP 3 as a CGI program, you may benchmark your build by typing **make bench**. Note that if safe mode is on by default, the benchmark may not be able to finish if it takes longer then the 30 seconds allowed. This is because the **set_time_limit()** can not be used in safe mode. Use the max_execution_time configuration setting to control this time for your own scripts. **make bench** ignores the configuration file.

> **Nota: make bench** is only available for PHP 3.

Using Variables

Some server supplied environment variables are not defined in the current CGI/1.1 specification. Only the following variables are defined there: *AUTH_TYPE*, *CONTENT_LENGTH*, *CONTENT_TYPE*, *GATEWAY_INTERFACE*, *PATH_INFO*, *PATH_TRANSLATED*,

QUERY_STRING, *REMOTE_ADDR*, *REMOTE_HOST*, *REMOTE_IDENT*, *REMOTE_USER*, *REQUEST_METHOD*, *SCRIPT_NAME*, *SERVER_NAME*, *SERVER_PORT*, *SERVER_PROTOCOL*, and *SERVER_SOFTWARE*. Everything else should be treated as 'vendor extensions'.

HP-UX specific installation notes

This section contains notes and hints specific to installing PHP on HP-UX systems. (Contributed by paul_mckay at clearwater-it dot co dot uk).

> **Nota:** These tips were written for PHP 4.0.4 and Apache 1.3.9.

1. You need gzip, download a binary distribution from `http://hpux.connect.org.uk/ftp/hpux/Gnu/gzip-1.2.4a/gzip-1.2.4a-sd-10.20.depot.Z` uncompress the file and install using swinstall.

2. You need gcc, download a binary distribution from `http://gatekeep.cs.utah.edu/ftp/hpux/Gnu/gcc-2.95.2/gcc-2.95.2-sd-10.20.depot.gz`. uncompress this file and install gcc using swinstall.

3. You need the GNU binutils, you can download a binary distribution from `http://hpux.connect.org.uk/ftp/hpux/Gnu/binutils-2.9.1/binutils-2.9.1-sd-10.20.depot.gz`. uncompress this file and install binutils using swinstall.

4. You now need bison, you can download a binary distribution from `http://hpux.connect.org.uk/ftp/hpux/Gnu/bison-1.28/bison-1.28-sd-10.20.depot.gz`, install as above.

5. You now need flex, you need to download the source from one of the http://www.gnu.org mirrors. It is in the non-gnu directory of the ftp site. Download the file, **gunzip**, then **tar -xvf** it. Go into the newly created flex directory and run **./configure**, followed by **make**, and then **make install**.

   If you have errors here, it's probably because gcc etc. are not in your PATH so add them to your PATH.

6. Download the PHP and apache sources.

7. **gunzip** and **tar -xvf** them. We need to hack a couple of files so that they can compile OK.

8. Firstly the configure file needs to be hacked because it seems to lose track of the fact that you are a hpux machine, there will be a better way of doing this but a cheap and cheerful hack is to put *lt_target=hpux10.20* on line 47286 of the configure script.

9. Next, the Apache GuessOS file needs to be hacked. Under `apache_1.3.9/src/helpers` change line 89 from *echo "hp${HPUXMACH}-hpux${HPUXVER}"; exit 0* to: *echo "hp${HPUXMACH}-hp-hpux${HPUXVER}"; exit 0*

10. You cannot install PHP as a shared object under HP-UX so you must compile it as a static, just follow the instructions at the Apache page.

11. PHP and Apache should have compiled OK, but Apache won't start. you need to create a new

user for Apache, e.g. www, or apache. You then change lines 252 and 253 of the `conf/httpd.conf` in Apache so that instead of

```
User nobody
Group nogroup
```

you have something like

```
User www
Group sys
```

This is because you can't run Apache as nobody under hp-ux. Apache and PHP should then work.

OpenBSD installation notes

This section contains notes and hints specific to installing PHP on OpenBSD 3.6.

Using Binary Packages

Using binary packages to install PHP on OpenBSD is the recommended and simplest method. The core package has been separated from the various modules, and each can be installed and removed independently from the others. The files you need can be found on your OpenBSD CD or on the FTP site.

The main package you need to install is `php4-core-4.3.8.tgz`, which contains the basic engine (plus gettext and iconv). Next, take a look at the module packages, such as `php4-mysql-4.3.8.tgz` or `php4-imap-4.3.8.tgz`. You need to use the **phpxs** command to activate and deactivate these modules in your `php.ini`.

Ejemplo 4-6. OpenBSD Package Install Example

```
# pkg_add php4-core-4.3.8.tgz
# /usr/local/sbin/phpxs -s
# cp /usr/local/share/doc/php4/php.ini-recommended /var/www/conf/php.ini
  (add in mysql)
# pkg_add php4-mysql-4.3.8.tgz
# /usr/local/sbin/phpxs -a mysql
  (add in imap)
# pkg_add php4-imap-4.3.8.tgz
# /usr/local/sbin/phpxs -a imap
  (remove mysql as a test)
# pkg_delete php4-mysql-4.3.8
# /usr/local/sbin/phpxs -r mysql
  (install the PEAR libraries)
# pkg_add php4-pear-4.3.8.tgz
```

Read the packages(7) manual page for more information about binary packages on OpenBSD.

Using Ports

You can also compile up PHP from source using the ports tree. However, this is only recommended for users familiar with OpenBSD. The PHP 4 port is split into two sub-directories: core and extensions. The extensions directory generates sub-packages for all of the supported PHP modules. If you find you do not want to create some of these modules, use the **no_*** FLAVOR. For example, to skip building the imap module, set the FLAVOR to **no_imap**.

Common Problems

- The default install of Apache runs inside a chroot(2) jail, which will restrict PHP scripts to accessing files under `/var/www`. You will therefore need to create a `/var/www/tmp` directory for PHP session files to be stored, or use an alternative session backend. In addition, database sockets need to be placed inside the jail or listen on the `localhost` interface. If you use network functions, some files from `/etc` such as `/etc/resolv.conf` and `/etc/services` will need to be moved into `/var/www/etc`. The OpenBSD PEAR package automatically installs into the correct chroot directories, so no special modification is needed there. More information on the OpenBSD Apache is available in the OpenBSD FAQ.

- The OpenBSD 3.6 package for the gd extension requires XFree86 to be installed. If you do not wish to use some of the font features that require X11, install the `php4-gd-4.3.8-no_x11.tgz` package instead.

Older Releases

Older releases of OpenBSD used the FLAVORS system to compile up a statically linked PHP. Since it is hard to generate binary packages using this method, it is now deprecated. You can still use the old stable ports trees if you wish, but they are unsupported by the OpenBSD team. If you have any comments about this, the current maintainer for the port is Anil Madhavapeddy (avsm at openbsd dot org).

Solaris specific installation tips

This section contains notes and hints specific to installing PHP on Solaris systems.

Required software

Solaris installs often lack C compilers and their related tools. Read this FAQ for information on why using GNU versions for some of these tools is necessary. The required software is as follows:

- gcc (recommended, other C compilers may work)
- make
- flex
- bison
- m4
- autoconf
- automake

- perl
- gzip
- tar
- GNU sed

In addition, you will need to install (and possibly compile) any additional software specific to your configuration, such as Oracle or MySQL.

Using Packages

You can simplify the Solaris install process by using pkgadd to install most of your needed components.

Gentoo installation notes

This section contains notes and hints specific to installing PHP on Gentoo Linux.

Using Portage (emerge)

While you can just download the PHP source and compile it yourself, using Gentoo's packaging system is the simplest and cleanest method of installing PHP. If you are not familiar with building software on Linux, this is the way to go.

If you have built your Gentoo system so far, you are probably used to Portage already. Installing Apache and PHP is no different than the other system tools.

The first decision you need to make is whether you want to install Apache 1.3.x or Apache 2.x. While both can be used with PHP, the steps given below will use Apache 1.3.x. Another thing to consider is whether your local Portage tree is up to date. If you have not updated it recently, you need to run **emerge sync** before anything else. This way, you will be using the most recent stable version of Apache and PHP.

Now that everything is in place, you can use the following example to install Apache and PHP:

Ejemplo 4-7. Gentoo Install Example with Apache 1.3

```
# emerge \<apache-2
# USE="-*" emerge php mod_php
# ebuild /var/db/pkg/dev-php/mod_php-<your PHP version>/mod_php-<your PHP version>.ebui
# nano /etc/conf.d/apache
  Add "-D PHP4" to APACHE_OPTS

# rc-update add apache default
# /etc/init.d/apache start
```

You can read more about emerge in the excellent Portage Manual provided on the Gentoo website.

If you need to use Apache 2, you can simply use **emerge apache** in the last example.

Better control on configuration

In the last section, PHP was emerged without any activated modules. As of this writing, the only module activated by default with Portage is XML which is needed by PEAR. This may not be what you want and you will soon discover that you need more activated modules, like MySQL, gettext, GD, etc.

When you compile PHP from source yourself, you need to activate modules via the **configure** command. With Gentoo, you can simply provide USE flags which will be passed to the configure script automatically. To see which USE flags to use with emerge, you can try:

Ejemplo 4-8. Getting the list of valid USE flags

```
# USE="-*" emerge -pv php

[ebuild  N    ] dev-php/php-4.3.6-r1  -X -berkdb -crypt -curl -debug -doc
-fdftk -firebird -flash -freetds -gd -gd-external -gdbm -gmp -hardenedphp
-imap -informix -ipv6 -java -jpeg -kerberos -ldap -mcal -memlimit -mssql
-mysql -ncurses -nls -oci8 -odbc -pam -pdflib -png -postgres -qt -readline
-snmp -spell -ssl -tiff -truetype -xml2 -yaz  3,876 kB
```

As you can see from the last output, PHP considers a lot of USE flags. Look at them closely and choose what you need. If you choose a flag and you do not have the proper libraries, Portage will compile them for you. It is a good idea to use **emerge -pv** again to see what Portage will compile in accordance to your USE flags. As an example, if you do not have X installed and you choose to include X in the USE flags, Portage will compile X prior to PHP, which can take a couple of hours.

If you choose to compile PHP with MySQL, cURL and GD support, the command will look something like this:

Ejemplo 4-9. Install PHP with USE flags

```
# USE="-* curl mysql gd" emerge php mod_php
```

As in the last example, do not forget to emerge php as well as mod_php. php is responsible for the command line version of PHP as mod_php is for the Apache module version of PHP.

Common Problems

- If you see the PHP source instead of the result the script should produce, you have probably forgot to edit `/etc/conf.d/apache`. Apache needs to be started with the -D PHP4 flag. To see if the flag is present, you should be able to see it when using **ps ax | grep apache** while Apache is running.

- Due to slotting problems, you might end up with more than one version of PHP installed on your system. If this is the case, you need to unmerge the old versions manually by using **emerge unmerge mod_php-<old version>**.

- If you cannot emerge PHP because of Java, try putting **-*** in front of your USE flags like in the above examples.

- If you are having problems configuring Apache and PHP, you can always search the Gentoo Forums. Try searching with the keywords "Apache PHP".

Capítulo 5. Installation on Mac OS X

This section contains notes and hints specific to installing PHP on Mac OS X. There are two slightly different versions of Mac OS X, Client and Server, our manual deals with installing PHP on both systems. Note that PHP is not available for MacOS 9 and earlier versions.

Using Packages

There are a few pre-packaged and pre-compiled versions of PHP for Mac OS X. This can help in setting up a standard configuration, but if you need to have a different set of features (such as a secure server, or a different database driver), you may need to build PHP and/or your web server yourself. If you are unfamiliar with building and compiling your own software, it's worth checking whether somebody has already built a packaged version of PHP with the features you need.

Compiling for OS X Server

Mac OS X Server install.

1. Get the latest distributions of Apache and PHP.

2. Untar them, and run the **configure** program on Apache like so.
```
./configure --exec-prefix=/usr \
--localstatedir=/var \
--mandir=/usr/share/man \
--libexecdir=/System/Library/Apache/Modules \
--iconsdir=/System/Library/Apache/Icons \
--includedir=/System/Library/Frameworks/Apache.framework/Versions/1.3/Headers \
--enable-shared=max \
--enable-module=most \
--target=apache
```
3. If you want the compiler to do some optimization, you may also want to add this line:
```
setenv OPTIM=-O2
```
4. Next, go to the PHP 4 source directory and configure it.
```
./configure --prefix=/usr \
    --sysconfdir=/etc \
    --localstatedir=/var \
    --mandir=/usr/share/man \
    --with-xml \
    --with-apache=/src/apache_1.3.12
```
   If you have any other additions (MySQL, GD, etc.), be sure to add them here. For the *--with-apache* string, put in the path to your apache source directory, for example `/src/apache_1.3.12`.

5. Type **make** and **make install**. This will add a directory to your Apache source directory under `src/modules/php4`.

6. Now, reconfigure Apache to build in PHP 4.

```
./configure --exec-prefix=/usr \
--localstatedir=/var \
--mandir=/usr/share/man \
--libexecdir=/System/Library/Apache/Modules \
--iconsdir=/System/Library/Apache/Icons \
--includedir=/System/Library/Frameworks/Apache.framework/Versions/1.3/Headers \
--enable-shared=max \
--enable-module=most \
--target=apache \
--activate-module=src/modules/php4/libphp4.a
```

You may get a message telling you that `libmodphp4.a` is out of date. If so, go to the `src/modules/php4` directory inside your Apache source directory and run this command: **ranlib libmodphp4.a**. Then go back to the root of the Apache source directory and run the above **configure** command again. That'll bring the link table up to date. Run **make** and **make install** again.

7. Copy and rename the `php.ini-dist` file to your `bin` directory from your PHP 4 source directory: `cp php.ini-dist /usr/local/bin/php.ini` or (if your don't have a local directory) `cp php.ini-dist /usr/bin/php.ini`.

Compiling for MacOS X Client

The following instructions will help you install a PHP module for the Apache web server included in MacOS X. This version includes support for the MySQL and PostgreSQL databases. These instructions are graciously provided by Marc Liyanage.

Aviso
Be careful when you do this, you could screw up your Apache web server!

Do this to install:

1. Open a terminal window.

2. Type `wget http://www.diax.ch/users/liyanage/software/macosx/libphp4.so.gz`, wait for the download to finish.

3. Type `gunzip libphp4.so.gz`.

4. Type `sudo apxs -i -a -n php4 libphp4.so`

5. Now type `sudo open -a TextEdit /etc/httpd/httpd.conf`. TextEdit will open with the web server configuration file. Locate these two lines towards the end of the file: (Use the Find command)

```
#AddType application/x-httpd-php .php
#AddType application/x-httpd-php-source .phps
```

Remove the two hash marks (#), then save the file and quit TextEdit.

6. Finally, type `sudo apachectl graceful` to restart the web server.

PHP should now be up and running. You can test it by dropping a file into your `Sites` folder which is called `test.php`. Into that file, write this line: *<?php phpinfo() ?>*.

Now open up *127.0.0.1/~your_username/test.php* in your web browser. You should see a status

table with information about the PHP module.

Capítulo 6. Installation on Windows systems

This section applies to Windows 98/Me and Windows NT/2000/XP/2003. PHP will not work on 16 bit platforms such as Windows 3.1 and sometimes we refer to the supported Windows platforms as Win32. Windows 95 is no longer supported as of PHP 4.3.0.

There are two main ways to install PHP for Windows: either manually or by using the installer.

If you have Microsoft Visual Studio, you can also build PHP from the original source code.

Once you have PHP installed on your Windows system, you may also want to load various extensions for added functionality.

Aviso
There are several all-in-one installers over the Internet, but none of those are endorsed by PHP.net, as we believe that the manual installation is the best choice to have your system secure and optimised.

Windows Installer

The Windows PHP installer is available from the downloads page at http://www.php.net/downloads.php. This installs the *CGI version* of PHP and for IIS, PWS, and Xitami, it configures the web server as well. The installer does not include any extra external PHP extensions (php_*.dll) as you'll only find those in the Windows Zip Package and PECL downloads.

> **Nota:** While the Windows installer is an easy way to make PHP work, it is restricted in many aspects as, for example, the automatic setup of extensions is not supported. Use of the installer isn't the preferred method for installing PHP.

First, install your selected HTTP (web) server on your system, and make sure that it works.

Run the executable installer and follow the instructions provided by the installation wizard. Two types of installation are supported - standard, which provides sensible defaults for all the settings it can, and advanced, which asks questions as it goes along.

The installation wizard gathers enough information to set up the `php.ini` file, and configure certain web servers to use PHP. One of the web servers the PHP installer does not configure for is Apache, so you'll need to configure it manually.

Once the installation has completed, the installer will inform you if you need to restart your system, restart the server, or just start using PHP.

Aviso
Be aware, that this setup of PHP is not secure. If you would like to have a secure PHP setup, you'd better go on the manual way, and set every option carefully. This automatically working setup gives you an instantly working PHP installation, but it is not meant to be used on online servers.

Manual Installation Steps

This install guide will help you manually install and configure PHP with a web server on Microsoft Windows. To get started you'll need to download the zip binary distribution from the downloads page at http://www.php.net/downloads.php.

Although there are many all-in-one installation kits, and we also distribute a PHP installer for Microsoft Windows, we recommend you take the time to setup PHP yourself as this will provide you with a better understanding of the system, and enables you to install PHP extensions easily when needed.

> **Upgrading from a previous PHP version:** Previous editions of the manual suggest moving various ini and DLL files into your SYSTEM (i.e. `C:\WINDOWS`) folder and while this simplifies the installation procedure it makes upgrading difficult. We advise you remove all of these files (like `php.ini` and PHP related DLLs from the Windows SYSTEM folder) before moving on with a new PHP installation. Be sure to backup these files as you might break the entire system. The old `php.ini` might be useful in setting up the new PHP as well. And as you'll soon learn, the preferred method for installing PHP is to keep all PHP related files in one directory and have this directory available to your systems PATH.
>
> **MDAC requirements:** If you use Microsoft *Windows 98/NT4* download the latest version of the Microsoft Data Access Components (MDAC) for your platform. MDAC is available at http://msdn.microsoft.com/data/. This requirement exists because ODBC is built into the distributed Windows binaries.

The following steps should be completed on all installations before any server specific instructions are performed:

Extract the distribution file into a directory of your choice. If you are installing PHP 4, extract to `C:\`, as the zip file expands to a foldername like `php-4.3.7-Win32`. If you are installing PHP 5, extract to `C:\php` as the zip file doesn't expand as in PHP 4. You may choose a different location but do not have spaces in the path (like `C:\Program Files\PHP`) as some web servers will crash if you do.

The directory structure extracted from the zip is different for PHP versions 4 and 5 and look like as follows:

Ejemplo 6-1. PHP 4 package structure

```
c:\php
   |
   +--cli
   |  |
   |  |-php.exe           -- CLI executable - ONLY for commandline scripting
   |
   +--dlls                -- support DLLs required by some extensions
   |  |
   |  |-expat.dll
   |  |
   |  |-fdftk.dll
   |  |
   |  |-...
   |
   +--extensions          -- extension DLLs for PHP
   |  |
   |  |-php_bz2.dll
   |  |
   |  |-php_cpdf.dll
   |  |
   |  |-..
   |
   +--mibs                -- support files for SNMP
   |
   +--openssl             -- support files for Openssl
   |
   +--pdf-related         -- support files for PDF
   |
   +--sapi                -- SAPI (server module support) DLLs
   |  |
   |  |-php4activescript.dll
   |  |
   |  |-php4apache.dll
   |  |
   |  |-php4apache2.dll
   |  |
   |  |-..
   |
   +--PEAR                -- initial copy of PEAR
   |
   |
   |-go-pear.bat          -- PEAR setup script
   |
   |-..
   |
   |-php.exe              -- CGI executable
   |
   |-..
   |
   |-php.ini-dist         -- default php.ini settings
   |
   |-php.ini-recommended  -- recommended php.ini settings
   |
   |-php4ts.dll           -- core PHP DLL
   |
   |-...
```

Or:

Ejemplo 6-2. PHP 5 package structure

```
c:\php
   |
   +--dev
   |  |
   |  |-php5ts.lib
   |
   +--ext                -- extension DLLs for PHP
   |  |
   |  |-php_bz2.dll
   |  |
   |  |-php_cpdf.dll
   |  |
   |  |-..
   |
   +--extras
   |  |
   |  +--mibs            -- support files for SNMP
   |  |
   |  +--openssl         -- support files for Openssl
   |  |
   |  +--pdf-related     -- support files for PDF
   |  |
   |  |-mime.magic
   |
   +--pear               -- initial copy of PEAR
   |
   |
   |-go-pear.bat         -- PEAR setup script
   |
   |-fdftk.dll
   |
   |-..
   |
   |-php-cgi.exe         -- CGI executable
   |
   |-php-win.exe         -- executes scripts without an opened command prompt
   |
   |-php.exe             -- CLI executable - ONLY for command line scripting
   |
   |-..
   |
   |-php.ini-dist        -- default php.ini settings
   |
   |-php.ini-recommended  -- recommended php.ini settings
   |
   |-php5activescript.dll
   |
   |-php5apache.dll
   |
   |-php5apache2.dll
   |
   |-..
   |
   |-php5ts.dll          -- core PHP DLL
   |
   |-...
```

Notice the differences and similarities. Both PHP 4 and PHP 5 have a CGI executable, a CLI executable, and server modules, but they are located in different folders and/or have different names. While PHP 4 packages have the server modules in the `sapi` folder, PHP 5 distributions have no such directory and instead they're in the PHP folder root. The supporting DLLs for the PHP 5 extensions are also not in a seperate directory.

> **Nota:** In PHP 4, you should move all files located in the `dll` and `sapi` folders to the main folder (e.g. `C:\php`).

Here is a list of server modules shipped with PHP 4 and PHP 5:

- `sapi/php4activescript.dll (php5activescript.dll)` - ActiveScript engine, allowing you to embed PHP in your Windows applications.
- `sapi/php4apache.dll (php5apache.dll)` - Apache 1.3.x module.
- `sapi/php4apache2.dll (php5apache2.dll)` - Apache 2.0.x module.
- `sapi/php4isapi.dll (php5isapi.dll)` - ISAPI Module for ISAPI compliant web servers like IIS 4.0/PWS 4.0 or newer.
- `sapi/php4nsapi.dll (php5nsapi.dll)` - Sun/iPlanet/Netscape server module.
- `sapi/php4pi3web.dll (no ñalent in PHP 5)` - Pi3Web server module.

Server modules provide significantly better performance and additional functionality compared to the CGI binary. The CLI version is designed to let you use PHP for command line scripting. More information about CLI is available in the chapter about using PHP from the command line.

Aviso
The SAPI modules have been significantly improved as of the 4.1 release, however, in older systems you may encounter server errors or other server modules failing, such as ASP.

The CGI and CLI binaries, and the web server modules all require the `php4ts.dll` (`php5ts.dll`) file to be available to them. You have to make sure that this file can be found by your PHP installation. The search order for this DLL is as follows:

- The same directory from where `php.exe` is called, or in case you use a SAPI module, the web server's directory (e.g. `C:\Program Files\Apache Group\Apache2\bin`).
- Any directory in your Windows *PATH* environment variable.

To make `php4ts.dll` / `php5ts.dll` available you have three options: copy the file to the Windows system directory, copy the file to the web server's directory, or add your PHP directory, `C:\php` to the *PATH*. For better maintenance, we advise you to follow the last option, add `C:\php` to the *PATH*, because it will be simpler to upgrade PHP in the future. Read more about how to add your PHP directory to *PATH* in the corresponding FAQ entry.

The next step is to set up a valid configuration file for PHP, `php.ini`. There are two ini files distributed in the zip file, `php.ini-dist` and `php.ini-recommended`. We advise you to use `php.ini-recommended`, because we optimized the default settings in this file for performance, and security. Read this well documented file carefully because it has changes from `php.ini-dist` that will drastically affect your setup. Some examples are display_errors being *off* and magic_quotes_gpc being *off*. In addition to reading these, study the ini settings and set every element manually yourself. If you would like to achieve the best security, then this is the way for you, although PHP works fine with these default ini files. Copy your chosen ini-file to a directory that PHP is able to find and rename it to `php.ini`. PHP searches for `php.ini` in the following locations (in order):

- PHPIniDir directive (Apache 2 module only)
- *HKEY_LOCAL_MACHINE\SOFTWARE\PHP\IniFilePath*
- The *PHPRC* environment variable

- Directory of PHP (for CLI), or the web server's directory (for SAPI modules)
- Windows directory (`C:\windows` or `C:\winnt`)

If you are running Apache 2, the simpler option is to use the PHPIniDir directive (read the installation on Apache 2 page), otherwise your best option is to set the *PHPRC* environment variable. This process is explained in the following FAQ entry.

> **Nota:** If you're using NTFS on Windows NT, 2000, XP or 2003, make sure that the user running the web server has read permissions to your `php.ini` (e.g. make it readable by Everyone).

The following steps are optional:

- Edit your new `php.ini` file. If you plan to use OmniHTTPd, do not follow the next step. Set the doc_root to point to your web servers document_root. For example:

```
doc_root = c:\inetpub\wwwroot // for IIS/PWS

doc_root = c:\apache\htdocs // for Apache
```

- Choose the extensions you would like to load when PHP starts. See the section about Windows extensions, about how to set up one, and what is already built in. Note that on a new installation it is advisable to first get PHP working and tested without any extensions before enabling them in `php.ini`.
- On PWS and IIS, you can set the browscap configuration setting to point to: `c:\windows\system\inetsrv\browscap.ini` on Windows 9x/Me, `c:\winnt\system32\inetsrv\browscap.ini` on NT/2000, and `c:\windows\system32\inetsrv\browscap.ini` on XP. For an up-to-date `browscap.ini`, read the following FAQ.

PHP is now setup on your system. The next step is to choose a web server, and enable it to run PHP. Choose a webserver from the table of contents.

ActiveScript

This section contains notes specific to the ActiveScript installation.

ActiveScript is a windows only SAPI that enables you to use PHP script in any ActiveScript compliant host, like Windows Script Host, ASP/ASP.NET, Windows Script Components or Microsoft Scriptlet control.

As of PHP 5.0.1, ActiveScript has been moved to the PECL repository. Podeis descargar esta DLL de las extensiones PECL desde la pagina PHP Downloads o desde http://snaps.php.net/.

> **Nota:** You should read the manual installation steps first!

After installing PHP, you should download the ActiveScript DLL (`php5activescript.dll`) and place it in the main PHP folder (e.g. `C:\php`).

After having all the files needed, you must register the DLL on your system. To achieve this, open a Command Prompt window (located in the Start Menu). Then go to your PHP directory by typing

something like *cd C:\php*. To register the DLL just type *regsvr32 php5activescript.dll*.

To test if ActiveScript is working, create a new file, named *test.wsf* (the extension is very important) and type:

```
<job id="test">

 <script language="PHPScript">
  $WScript->Echo("Hello World!");
 </script>

</job>
```

Save and double-click on the file. If you receive a little window saying "Hello World!" you're done.

> **Nota:** ActiveScript doesn't use the default `php.ini` file. Instead, it will look only in the same directory as the .exe that caused it to load. You should create `php-activescript.ini` and place it in that folder, if you wish to load extensions, etc.

Microsoft IIS / PWS

This section contains notes and hints specific to IIS (Microsoft Internet Information Server). We have included installation instructions for PWS/IIS 3, PWS 4 or newer and IIS 4 or newer versions.

> **Important for CGI users:** Read the faq on cgi.force_redirect for important details. This directive needs to be set to *0*.

Aviso
Al usar el modo CGI, vuestro servidor esta expuesto a diferentes ataques. Por favor, leer la sección Seguridad con CGI para aprender como defenderse de estos ataques.

Windows and PWS/IIS 3

The recommended method for configuring these servers is to use the REG file included with the distribution (`pws-php4cgi.reg` in the SAPI folder for PHP 4, or `pws-php5cgi.reg` in the main folder for PHP 5). You may want to edit this file and make sure the extensions and PHP install directories match your configuration. Or you can follow the steps below to do it manually.

Aviso
These steps involve working directly with the Windows registry. One error here can leave your system in an unstable state. We highly recommend that you back up your registry first. The PHP Development team will not be held responsible if you damage your registry.

- Run Regedit.
- Navigate to: *HKEY_LOCAL_MACHINE /System /CurrentControlSet /Services /W3Svc / Parameters /ScriptMap*.
- On the edit menu select: *New->String Value*.
- Type in the extension you wish to use for your php scripts. For example *.php*
- Double click on the new string value and enter the path to `php.exe` in the value data field.

ex: `C:\php\php.exe` for PHP 4, or `C:\php\php-cgi.exe` for PHP 5.

- Repeat these steps for each extension you wish to associate with PHP scripts.

The following steps do not affect the web server installation and only apply if you want your PHP scripts to be executed when they are run from the command line (ex. run `C:\myscripts\test.php`) or by double clicking on them in a directory viewer window. You may wish to skip these steps as you might prefer the PHP files to load into a text editor when you double click on them.

- Navigate to: *HKEY_CLASSES_ROOT*
- On the edit menu select: *New->Key*.
- Name the key to the extension you setup in the previous section. ex: *.php*
- Highlight the new key and in the right side pane, double click the "default value" and enter *phpfile*.
- Repeat the last step for each extension you set up in the previous section.
- Now create another *New->Key* under *HKEY_CLASSES_ROOT* and name it *phpfile*.
- Highlight the new key *phpfile* and in the right side pane, double click the "default value" and enter *PHP Script*.
- Right click on the *phpfile* key and select *New->Key*, name it *Shell*.
- Right click on the *Shell* key and select *New->Key*, name it *open*.
- Right click on the *open* key and select *New->Key*, name it *command*.
- Highlight the new key *command* and in the right side pane, double click the "default value" and enter the path to `php.exe`. ex: *c:\php\php.exe -q %1*. (don't forget the *%1*).
- Exit Regedit.
- If using PWS on Windows, reboot to reload the registry.

PWS and IIS 3 users now have a fully operational system. IIS 3 users can use a nifty tool from Steven Genusa to configure their script maps.

Windows and PWS 4 or newer

When installing PHP on Windows with PWS 4 or newer version, you have two options. One to set up the PHP CGI binary, the other is to use the ISAPI module DLL.

If you choose the CGI binary, do the following:

- Edit the enclosed `pws-php4cgi.reg` / `pws-php5cgi.reg` file (look into the SAPI folder for PHP 4, or in the main folder for PHP 5) to reflect the location of your `php.exe` / `php-cgi.exe`. Backslashes should be escaped, for example:

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\w3svc\parameters\Script Map] ".php"="C:\\php\\php.exe" (change to *C:\\php\\php-cgi.exe* if you are using PHP 5) Now merge this registery file into your system; you may do this by double-clicking it.

- In the PWS Manager, right click on a given directory you want to add PHP support to, and select Properties. Check the 'Execute' checkbox, and confirm.

If you choose the ISAPI module, do the following:

- Edit the enclosed `pws-php4isapi.reg` / `pws-php5isapi.reg` file (look into the SAPI folder for PHP 4, or in the main folder for PHP 5) to reflect the location of your `php4isapi.dll` / `php5isapi.dll`. Backslashes should be escaped, for example: *[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\w3svc\parameters\Script Map] ".php"="C:\\php\\sapi\\php4isapi.dll"* (or *C:\\php\\php5isapi.dll* for PHP 5) Now merge this registery file into your system; you may do this by double-clicking it.

- In the PWS Manager, right click on a given directory you want to add PHP support to, and select Properties. Check the 'Execute' checkbox, and confirm.

Windows NT/2000/XP and IIS 4 or newer

To install PHP on an NT/2000/XP Server running IIS 4 or newer, follow these instructions. You have two options to set up PHP, using the CGI binary (`php.exe` in PHP 4, or `php-cgi.exe` in PHP 5) or with the ISAPI module.

In either case, you need to start the Microsoft Management Console (may appear as 'Internet Services Manager', either in your Windows NT 4.0 Option Pack branch or the Control Panel=>Administrative Tools under Windows 2000/XP). Then right click on your Web server node (this will most probably appear as 'Default Web Server'), and select 'Properties'.

If you want to use the CGI binary, do the following:

- Under 'Home Directory', 'Virtual Directory', or 'Directory', click on the 'Configuration' button, and then enter the App Mappings tab.

- Click Add, and in the Executable box, type: *C:\php\php.exe* for PHP 4 or *C:\php\php-cgi.exe* for PHP 5 (assuming that you have unziped PHP in `c:\php\`).

- In the Extension box, type the file name extension you want associated with PHP scripts. Leave 'Method exclusions' blank, and check the 'Script engine' checkbox. You may also like to check the 'check that file exists' box - for a small performance penalty, IIS (or PWS) will check that the script file exists and sort out authentication before firing up PHP. This means that you will get sensible 404 style error messages instead of CGI errors complaining that PHP did not output any data.

  You must start over from the previous step for each extension you want associated with PHP scripts. *.php* and *.phtml* are common, although *.php3* may be required for legacy applications.

- Set up the appropriate security. (This is done in Internet Service Manager), and if your NT Server uses NTFS file system, add execute rights for I_USR_ to the directory that contains `php.exe` / `php-cgi.exe`.

To use the ISAPI module, do the following:

- If you don't want to perform HTTP Authentication using PHP, you can (and should) skip this step. Under ISAPI Filters, add a new ISAPI filter. Use PHP as the filter name, and supply a path to the `php4isapi.dll` / `php5isapi.dll`.
- Under 'Home Directory', click on the 'Configuration' button. Add a new entry to the Application Mappings. Use the path to the `php4isapi.dll` / `php5isapi.dll` as the Executable, supply *.php* as the extension, leave 'Method exclusions' blank, and check the 'Script engine' checkbox.
- Stop IIS completely (NET STOP iisadmin)
- Start IIS again (NET START w3svc)

If you experience 100% CPU usage after some time, turn off the IIS setting *Cache ISAPI Application*.

Apache 1.3.x on Microsoft Windows

This section contains notes and hints specific to Apache 1.3.x installs of PHP on Microsoft Windows systems. There are also instructions and notes for Apache 2 on a separate page.

> **Nota:** Please read the manual installation steps first!

There are two ways to set up PHP to work with Apache 1.3.x on Windows. One is to use the CGI binary (`php.exe` for PHP 4 and `php-cgi.exe` for PHP 5), the other is to use the Apache Module DLL. In either case you need to edit your `httpd.conf` to configure Apache to work with PHP, and then restart the server.

It is worth noting here that now the SAPI module has been made more stable under Windows, we recommend it's use above the CGI binary, since it is more transparent and secure.

Although there can be a few variations of configuring PHP under Apache, these are simple enough to be used by the newcomer. Please consult the Apache Documentation for further configuration directives.

After changing the configuration file, remember to restart the server, for example, **NET STOP APACHE** followed by **NET START APACHE**, if you run Apache as a Windows Service, or use your regular shortcuts.

> **Nota:** Recordar que cuando utiliceis valores de PATH en Windows con barras invertidas tal como `c:\directory\file.ext`, estas deben de ser convertidas a barras inclinadas, `c:/directory/file.ext`.

Installing as an Apache module

You should add the following lines to your Apache `httpd.conf` file:

Ejemplo 6-3. PHP as an Apache 1.3.x module

This assumes PHP is installed to `c:\php`. Adjust the path if this is not the case.

For PHP 4:

```
# Add to the end of the LoadModule section
LoadModule php4_module "c:/php/php4apache.dll"

# Add to the end of the AddModule section
AddModule mod_php4.c
```

For PHP 5:

```
# Add to the end of the LoadModule section
LoadModule php5_module "c:/php/php5apache.dll"

# Add to the end of the AddModule section
AddModule mod_php5.c
```

For both:

```
# Add this line inside the <IfModule mod_mime.c> conditional brace
AddType application/x-httpd-php .php

# For syntax highlighted .phps files, also add
AddType application/x-httpd-php-source .phps
```

Installing as a CGI binary

If you unzipped the PHP package to `C:\php\` as described in the Manual Installation Steps section, you need to insert these lines to your Apache configuration file to set up the CGI binary:

Ejemplo 6-4. PHP and Apache 1.3.x as CGI

```
ScriptAlias /php/ "c:/php/"
AddType application/x-httpd-php .php

# For PHP 4
Action application/x-httpd-php "/php/php.exe"

# For PHP 5
Action application/x-httpd-php "/php/php-cgi.exe"

# specify the directory where php.ini is
SetEnv PHPRC C:/php
```

Note that the second line in the list above can be found in the actual versions of `httpd.conf`, but it is commented out. Remember also to substitute the `c:/php/` for your actual path to PHP.

Aviso
Al usar el modo CGI, vuestro servidor esta expuesto a diferentes ataques. Por favor, leer la sección Seguridad con CGI para aprender como defenderse de estos ataques.

If you would like to present PHP source files syntax highlighted, there is no such convenient option as with the module version of PHP. If you chose to configure Apache to use PHP as a CGI binary, you will need to use the **highlight_file()** function. To do this simply create a PHP script file and add this code: *<?php highlight_file('some_php_script.php'); ?>*.

Apache 2.0.x on Microsoft Windows

This section contains notes and hints specific to Apache 2.0.x installs of PHP on Microsoft Windows systems. We also have instructions and notes for Apache 1.3.x users on a separate page.

Nota: You should read the manual installation steps first!

Aviso
No usar Apache 2.0 y PHP en un sistema en produccion, tanto en Unix como en Windows. Consultar la FAQ

You are highly encouraged to take a look at the Apache Documentation to get a basic understanding of the Apache 2.0.x Server. Also consider to read the Windows specific notes for Apache 2.0.x before reading on here.

PHP and Apache 2.0.x compatibility notes: The following versions of PHP are known to work with the most recent version of Apache 2.0.x:

- PHP 4.3.0 or later available at http://www.php.net/downloads.php.
- the latest stable development version. Get the source code http://snaps.php.net/php4-latest.tar.gz or download binaries for Windows http://snaps.php.net/win32/php4-win32-latest.zip.
- a prerelease version downloadable from http://qa.php.net/.
- you have always the option to obtain PHP through anonymous CVS.

These versions of PHP are compatible to Apache 2.0.40 and later.

Apache 2.0 *SAPI*-support started with PHP 4.2.0. PHP 4.2.3 works with Apache 2.0.39, don't use any other version of Apache with PHP 4.2.3. However, the recommended setup is to use PHP 4.3.0 or later with the most recent version of Apache2.

All mentioned versions of PHP will work still with Apache 1.3.x.

Aviso
Apache 2.0.x is designed to run on Windows NT 4.0, Windows 2000 or Windows XP. At this time, support for Windows 9x is incomplete. Apache 2.0.x is not expected to work on those platforms at this time.

Download the most recent version of Apache 2.0.x and a fitting PHP version. Follow the Manual Installation Steps and come back to go on with the integration of PHP and Apache.

There are two ways to set up PHP to work with Apache 2.0.x on Windows. One is to use the CGI binary the other is to use the Apache module DLL. In either case you need to edit your `httpd.conf` to configure Apache to work with PHP and then restart the server.

Nota: Recordar que cuando utiliceis valores de PATH en Windows con barras invertidas tal como `c:\directory\file.ext`, estas deben de ser convertidas a barras inclinadas, `c:/directory/file.ext`.

Installing as a CGI binary

You need to insert these three lines to your Apache `httpd.conf` configuration file to set up the CGI binary:

Ejemplo 6-5. PHP and Apache 2.0 as CGI

```
ScriptAlias /php/ "c:/php/"
AddType application/x-httpd-php .php

# For PHP 4
Action application/x-httpd-php "/php/php.exe"

# For PHP 5
Action application/x-httpd-php "/php/php-cgi.exe"
```

Aviso
Al usar el modo CGI, vuestro servidor esta expuesto a diferentes ataques. Por favor, leer la sección Seguridad con CGI para aprender como defenderse de estos ataques.

Installing as an Apache module

You need to insert these two lines to your Apache `httpd.conf` configuration file to set up the PHP module for Apache 2.0:

Ejemplo 6-6. PHP and Apache 2.0 as Module

```
# For PHP 4 do something like this:
LoadModule php4_module "c:/php/php4apache2.dll"
AddType application/x-httpd-php .php

# For PHP 5 do something like this:
LoadModule php5_module "c:/php/php5apache2.dll"
AddType application/x-httpd-php .php

# configure the path to php.ini
PHPIniDir "C:/php"
```

> **Nota:** Remember to substitute the `c:/php/` for your actual path to PHP in the above examples. Take care to use either `php4apache2.dll` or `php5apache2.dll` in your LoadModule directive and *not* `php4apache.dll` or `php5apache.dll` as the latter ones are designed to run with Apache 1.3.x.
>
> **Nota:** If you want to use content negotiation, read related FAQ.

Aviso
Don't mix up your installation with DLL files from *different PHP versions*. You have the only choice to use the DLL's and extensions that ship with your downloaded PHP version.

Sun, iPlanet and Netscape servers on Microsoft Windows

This section contains notes and hints specific to Sun Java System Web Server, Sun ONE Web Server, iPlanet and Netscape server installs of PHP on Windows.

From PHP 4.3.3 on you can use PHP scripts with the NSAPI module to generate custom directory listings and error pages. Additional functions for Apache compatibility are also available. For support in current webservers read the note about subrequests.

CGI setup on Sun, iPlanet and Netscape servers

To install PHP as a CGI handler, do the following:

- Copy `php4ts.dll` to your systemroot (the directory where you installed Windows)
- Make a file association from the command line. Type the following two lines:
```
assoc .php=PHPScript
ftype PHPScript=c:\php\php.exe %1 %*
```
- In the Netscape Enterprise Administration Server create a dummy shellcgi directory and remove it just after (this step creates 5 important lines in obj.conf and allow the web server to handle shellcgi scripts).
- In the Netscape Enterprise Administration Server create a new mime type (Category: type, Content-Type: magnus-internal/shellcgi, File Suffix:php).
- Do it for each web server instance you want PHP to run

More details about setting up PHP as a CGI executable can be found here:
http://benoit.noss.free.fr/php/install-php.html

NSAPI setup on Sun, iPlanet and Netscape servers

To install PHP with NSAPI, do the following:

- Copy `php4ts.dll` to your systemroot (the directory where you installed Windows)
- Make a file association from the command line. Type the following two lines:
```
assoc .php=PHPScript
ftype PHPScript=c:\php\php.exe %1 %*
```
- In the Netscape Enterprise Administration Server create a new mime type (Category: type, Content-Type: magnus-internal/x-httpd-php, File Suffix: php).
- Edit `magnus.conf` (for servers >= 6) or `obj.conf` (for servers < 6) and add the following: You should place the lines after *mime types init*.
```
Init fn="load-modules" funcs="php4_init,php4_execute,php4_auth_trans" shlib="c:/p
Init fn="php4_init" LateInit="yes" errorString="Failed to initialise PHP!" [php_i
```
(PHP >= 4.3.3) The *php_ini* parameter is optional but with it you can place your `php.ini` in your webserver config directory.
- Configure the default object in `obj.conf` (for virtual server classes [Sun Web Server 6.0+] in their `vserver.obj.conf`): In the <*Object name="default"*> section, place this line necessarily after all 'ObjectType' and before all 'AddLog' lines:
```
Service fn="php4_execute" type="magnus-internal/x-httpd-php" [inikey=value inikey
```
(PHP >= 4.3.3) As additional parameters you can add some special `php.ini`-values, for example you can set a *docroot="/path/to/docroot"* specific to the context *php4_execute* is called. For boolean ini-keys please use 0/1 as value, not *"On","Off",...* (this will not work correctly), e.g. *zlib.output_compression=1* instead of *zlib.output_compression="On"*
- This is only needed if you want to configure a directory that only consists of PHP scripts (same like a cgi-bin directory):
```
<Object name="x-httpd-php">
ObjectType fn="force-type" type="magnus-internal/x-httpd-php"
Service fn=php4_execute [inikey=value inikey=value ...]
</Object>
```
After that you can configure a directory in the Administration server and assign it the style *x-*

httpd-php. All files in it will get executed as PHP. This is nice to hide PHP usage by renaming files to `.html`.

- Restart your web service and apply changes
- Do it for each web server instance you want PHP to run

Nota: More details about setting up PHP as an NSAPI filter can be found here: http://benoit.noss.free.fr/php/install-php4.html

Nota: The stacksize that PHP uses depends on the configuration of the webserver. If you get crashes with very large PHP scripts, it is recommended to raise it with the Admin Server (in the section "MAGNUS EDITOR").

CGI environment and recommended modifications in php.ini

Important when writing PHP scripts is the fact that Sun JSWS/Sun ONE WS/iPlanet/Netscape is a multithreaded web server. Because of that all requests are running in the same process space (the space of the webserver itself) and this space has only one environment. If you want to get CGI variables like *PATH_INFO*, *HTTP_HOST* etc. it is not the correct way to try this in the old PHP 3.x way with **getenv()** or a similar way (register globals to environment, *$_ENV*). You would only get the environment of the running webserver without any valid CGI variables!

Nota: Why are there (invalid) CGI variables in the environment?

Answer: This is because you started the webserver process from the admin server which runs the startup script of the webserver, you wanted to start, as a CGI script (a CGI script inside of the admin server!). This is why the environment of the started webserver has some CGI environment variables in it. You can test this by starting the webserver not from the administration server. Use the command line as root user and start it manually - you will see there are no CGI-like environment variables.

Simply change your scripts to get CGI variables in the correct way for PHP 4.x by using the superglobal *$_SERVER*. If you have older scripts which use *$HTTP_HOST*, etc., you should turn on *register_globals* in `php.ini` and change the variable order too (important: remove *"E"* from it, because you do not need the environment here):

```
variables_order = "GPCS"
register_globals = On
```

Special use for error pages or self-made directory listings (PHP >= 4.3.3)

You can use PHP to generate the error pages for *"404 Not Found"* or similar. Add the following line to the object in `obj.conf` for every error page you want to overwrite:

```
Error fn="php4_execute" code=XXX script="/path/to/script.php" [inikey=value inikey=valu
```

where *XXX* is the HTTP error code. Please delete any other *Error* directives which could interfere with yours. If you want to place a page for all errors that could exist, leave the *code* parameter out. Your script can get the HTTP status code with *$_SERVER['ERROR_TYPE']*.

Another possibility is to generate self-made directory listings. Just create a PHP script which displays a directory listing and replace the corresponding default Service line for *type="magnus-internal/directory"* in `obj.conf` with the following:

```
Service fn="php4_execute" type="magnus-internal/directory" script="/path/to/script.php"
```
For both error and directory listing pages the original URI and translated URI are in the variables *$_SERVER['PATH_INFO']* and *$_SERVER['PATH_TRANSLATED']*.

Note about nsapi_virtual() and subrequests (PHP >= 4.3.3)

The NSAPI module now supports the **nsapi_virtual()** function (alias: **virtual()**) to make subrequests on the webserver and insert the result in the webpage. The problem is, that this function uses some undocumented features from the NSAPI library.

Under Unix this is not a problem, because the module automatically looks for the needed functions and uses them if available. If not, **nsapi_virtual()** is disabled.

Under Windows limitations in the DLL handling need the use of a automatic detection of the most recent `ns-httpdXX.dll` file. This is tested for servers till version 6.1. If a newer version of the Sun server is used, the detection fails and **nsapi_virtual()** is disabled.

If this is the case, try the following: Add the following parameter to *php4_init* in `magnus.conf`/`obj.conf`:
```
Init fn=php4_init ... server_lib="ns-httpdXX.dll"
```
where *XX* is the correct DLL version number. To get it, look in the server-root for the correct DLL name. The DLL with the biggest filesize is the right one.

You can check the status by using the **phpinfo()** function.

> **Nota:** But be warned: Support for **nsapi_virtual()** is EXPERIMENTAL!!!

OmniHTTPd Server

This section contains notes and hints specific to OmniHTTPd on Windows.

> **Nota:** You should read the manual installation steps first!

Aviso
Al usar el modo CGI, vuestro servidor esta expuesto a diferentes ataques. Por favor, leer la sección Seguridad con CGI para aprender como defenderse de estos ataques.

You need to complete the following steps to make PHP work with OmniHTTPd. This is a CGI executable setup. SAPI is supported by OmniHTTPd, but some tests have shown that it is not so stable to use PHP as an ISAPI module.

> **Important for CGI users:** Read the faq on cgi.force_redirect for important details. This directive needs to be set to *0*.

1. Install OmniHTTPd server.
2. Right click on the blue OmniHTTPd icon in the system tray and select *Properties*
3. Click on *Web Server Global Settings*

4. On the 'External' tab, enter: *virtual* = *.php* | *actual* = *c:\php\php.exe* (use `php-cgi.exe` if installing PHP 5), and use the Add button.

5. On the *Mime* tab, enter: *virtual* = *wwwserver/stdcgi* | *actual* = *.php*, and use the Add button.

6. Click *OK*

Repeat steps 2 - 6 for each extension you want to associate with PHP.

> **Nota:** Some OmniHTTPd packages come with built in PHP support. You can choose at setup time to do a custom setup, and uncheck the PHP component. We recommend you to use the latest PHP binaries. Some OmniHTTPd servers come with PHP 4 beta distributions, so you should choose not to set up the built in support, but install your own. If the server is already on your machine, use the Replace button in Step 4 and 5 to set the new, correct information.

Sambar Server on Microsoft Windows

This section contains notes and hints specific to the Sambar Server for Windows.

> **Nota:** You should read the manual installation steps first!

This list describes how to set up the ISAPI module to work with the Sambar server on Windows.

- Find the file called `mappings.ini` (in the config directory) in the Sambar install directory.

- Open `mappings.ini` and add the following line under *[ISAPI]*:
  Ejemplo 6-7. ISAPI configuration of Sambar
  ```
  #for PHP 4
  *.php = c:\php\php4isapi.dll

  #for PHP 5
  *.php = c:\php\php5isapi.dll
  ```
  (This line assumes that PHP was installed in `c:\php`.)

- Now restart the Sambar server for the changes to take effect.

Xitami on Microsoft Windows

This section contains notes and hints specific to Xitami on Windows.

> **Nota:** You should read the manual installation steps first!

This list describes how to set up the PHP CGI binary to work with Xitami on Windows.

> **Important for CGI users:** Read the faq on cgi.force_redirect for important details. This directive needs to be set to *0*. If you want to use *$_SERVER['PHP_SELF']* you have to enable the cgi.fix_pathinfo directive.

Aviso
Al usar el modo CGI, vuestro servidor esta expuesto a diferentes ataques. Por favor, leer la sección Seguridad con CGI para aprender como defenderse de estos ataques.

- Make sure the webserver is running, and point your browser to xitamis admin console (usually *http://127.0.0.1/admin*), and click on Configuration.
- Navigate to the Filters, and put the extension which PHP should parse (i.e. .php) into the field File extensions (.xxx).
- In Filter command or script put the path and name of your PHP CGI executable i.e. `C:\php\php.exe` for PHP 4, or `C:\php\php-cgi.exe` for PHP 5.
- Press the 'Save' icon.
- Restart the server to reflect changes.

Building from source

Before getting started, it is worthwhile answering the question: "Why is building on Windows so hard?" Two reasons come to mind:

1. Windows does not (yet) enjoy a large community of developers who are willing to freely share their source. As a direct result, the necessary investment in infrastructure required to support such development hasn't been made. By and large, what is available has been made possible by the porting of necessary utilities from Unix. Don't be surprised if some of this heritage shows through from time to time.
2. Pretty much all of the instructions that follow are of the "set and forget" variety. So sit back and try follow the instructions below as faithfully as you can.

Requirimientos

To compile and build PHP you need a Microsoft Development Environment. Microsoft Visual C++ 6.0 is recommended. To extract the downloaded files you need a extraction utility (e.g.: Winzip). If you don't already have an unzip utility, you can get a free version from InfoZip.

Before you get started, you have to download...

- ..the win32 buildtools from the PHP site at http://www.php.net/extra/win32build.zip.
- ..the source code for the DNS name resolver used by PHP from http://www.php.net/extra/bindlib_w32.zip. This is a replacement for the `resolv.lib` library included in `win32build.zip`.
- If you plan to compile PHP as a Apache module you will also need the Apache sources.

Finally, you are going to need the source to PHP itself. You can get the latest development version using anonymous CVS, a snapshot or the most recent released source tarball.

Putting it all together

After downloading the required packages you have to extract them in a proper place.

- Create a working directory where all files end up after extracting, e.g: `C:\work`.
- Create the directory `win32build` under your working directory (`C:\work`) and unzip `win32build.zip` into it.
- Create the directory `bindlib_w32` under your working directory (`C:\work`) and unzip `bindlib_w32.zip` into it.
- Extract the downloaded PHP source code into your working directory (`C:\work`).

Following this steps your directory structure looks like this:

```
+--c:\work
|   |
|   +--bindlib_w32
|   |   |
|   |   +--arpa
|   |   |
|   |   +--conf
|   |   |
|   |   +--...
|   |
|   +--php-4.x.x
|   |   |
|   |   +--build
|   |   |
|   |   +--...
|   |   |
|   |   +--win32
|   |   |
|   |   +--...
|   |
|   +--win32build
|   |   |
|   |   +--bin
|   |   |
|   |   +--include
|   |   |
|   |   +--lib
```

Create the directories `c:\usr\local\lib`. Copy `bison.simple` from `c:\work\win32build\bin` to `c:\usr\local\lib`.

> **Nota:** Cygwin users may omit the last step. A properly installed Cygwin environment provides the mandatory files `bison.simple` and `bison.exe`.

Configure MVC ++

The next step is to configure MVC ++ to prepare for compiling. Launch Microsoft Visual C++, and from the menu select Tools => Options. In the dialog, select the directories tab. Sequentially change the dropdown to Executables, Includes, and Library files. Your entries should look like this:

- Executable files: `c:\work\win32build\bin`, Cygwin users: `cygwin\bin`
- Include files: `c:\work\win32build\include`

- Library files: `c:\work\win32build\lib`

Build resolv.lib

You must build the `resolv.lib` library. Decide whether you want to have debug symbols available (bindlib - Win32 Debug) or not (bindlib - Win32 Release). Build the appropriate configuration:

- For GUI users, launch VC++, and then select File => Open Workspace, navigate to `c:\work\bindlib_w32` and select `bindlib.dsw`. Then select Build=>Set Active Configuration and select the desired configuration. Finally select Build=>Rebuild All.
- For command line users, make sure that you either have the C++ environment variables registered, or have run **vcvars.bat**, and then execute one of the following commands:
  - `msdev bindlib.dsp /MAKE "bindlib - Win32 Debug"`
  - `msdev bindlib.dsp /MAKE "bindlib - Win32 Release"`

At this point, you should have a usable `resolv.lib` in either your `c:\work\bindlib_w32\Debug` or `Release` subdirectories. Copy this file into your `c:\work\win32build\lib` directory over the file by the same name found in there.

Compiling

The best way to get started is to build the CGI version.

- For GUI users, launch VC++, and then select File => Open Workspace and select `c:\work\php-4.x.x\win32\php4ts.dsw` . Then select Build=>Set Active Configuration and select the desired configuration, either *php4ts - Win32 Debug_TS* or *php4ts - Win32 Release_TS*. Finally select Build=>Rebuild All.
- For command line users, make sure that you either have the C++ environment variables registered, or have run **vcvars.bat**, and then execute one of the following commands from the `c:\work\php-4.x.x\win32` directory:
  - `msdev php4ts.dsp /MAKE "php4ts - Win32 Debug_TS"`
  - `msdev php4ts.dsp /MAKE "php4ts - Win32 Release_TS"`
  - At this point, you should have a usable `php.exe` in either your `c:\work\php-4.x.x.\Debug_TS` or `Release_TS` subdirectories.

It is possible to do minor customization to the build process by editing the `main/config.win32.h` file. For example you can change the default location of `php.ini`, the built-in extensions, and the default location for your extensions.

Next you may want to build the CLI version which is designed to use PHP from the command line. The steps are the same as for building the CGI version, except you have to select the *php4ts_cli - Win32 Debug_TS* or *php4ts_cli - Win32 Release_TS* project file. After a successful compiling run

you will find the `php.exe` in either the directory `Release_TS\cli\` or `Debug_TS\cli\`.

> **Nota:** If you want to use PEAR and the comfortable command line installer, the CLI-SAPI is mandatory. For more information about PEAR and the installer read the documentation at the PEAR website.

In order to build the SAPI module (`php4isapi.dll`) for integrating PHP with Microsoft IIS, set your active configuration to `php4isapi-whatever-config` and build the desired dll.

Installation of extensions on Windows

After installing PHP and a webserver on Windows, you will probably want to install some extensions for added functionality. You can choose which extensions you would like to load when PHP starts by modifying your `php.ini`. You can also load a module dynamically in your script using **dl()**.

The DLLs for PHP extensions are prefixed with *php_*.

> **Nota:** In PHP 4.3.1 BCMath, Calendar, COM, Ctype, FTP, MySQL, ODBC, Overload, PCRE, Session, Tokenizer, WDDX, XML and Zlib support is *built in*. You don't need to load any additional extensions in order to use these functions. See your distributions `README.txt` or `install.txt` or this table for a list of built in modules.

The default location PHP searches for extensions is `c:\php4\extensions` in PHP 4 and `c:\php5` in PHP 5. To change this setting to reflect your setup of PHP edit your `php.ini` file:

- You will need to change the extension_dir setting to point to the directory where your extensions lives, or where you have placed your `php_*.dll` files. Please do not forget the last backslash. For example:

```
extension_dir = c:/php/extensions/
```

- Enable the extension(s) in `php.ini` you want to use by uncommenting the *extension=php_*.dll* lines in `php.ini`. This is done by deleting the leading ; form the extension you want to load.
  Ejemplo 6-8. Enable Bzip2 extension for PHP-Windows

```
// change the following line from ...
;extension=php_bz2.dll

// ... to
extension=php_bz2.dll
```

- Some of the extensions need extra DLLs to work. Couple of them can be found in the distribution package, in the `C:\php\dlls\` folder in PHP 4 or in the main folder in PHP 5, but some, for example Oracle (`php_oci8.dll`) require DLLs which are not bundled with the distribution package. If you are installing PHP 4, copy the bundled DLLs from `C:\php\dlls` folder to the main `C:\php` folder. Don't forget to include `C:\php` in the system *PATH* (this process is explained in a separate FAQ entry).

- Some of these DLLs are not bundled with the PHP distribution. See each extensions documentation page for details. Also, read the manual section titled Installation of PECL extensions for details on PECL. An increasingly large number of PHP extensions are found in PECL, and these extensions require a separate download.

Nota: If you are running a server module version of PHP remember to restart your webserver to reflect your changes to `php.ini`.

The following table describes some of the extensions available and required additional dlls.

Tabla 6-1. PHP Extensions

Extension	Description	Notes
php_bz2.dll	bzip2 compression functions	None
php_calendar.dll	Calendar conversion functions	Built in since PHP 4.0.3
php_cpdf.dll	ClibPDF functions	None
php_crack.dll	Crack functions	None
php_ctype.dll	ctype family functions	Built in since PHP 4.3.0
php_curl.dll	CURL, Client URL library functions	Requires: `libeay32.dll`, `ssleay32.dll` (bundled)
php_cybercash.dll	Cybercash payment functions	PHP <= 4.2.0
php_db.dll	DBM functions	Deprecated. Use DBA instead (`php_dba.dll`)
php_dba.dll	DBA: DataBase (dbm-style) Abstraction layer functions	None
php_dbase.dll	dBase functions	None
php_dbx.dll	dbx functions	
php_domxml.dll	DOM XML functions	PHP <= 4.2.0 requires: `libxml2.dll` (bundled) PHP >= 4.3.0 requires: `iconv.dll` (bundled)
php_dotnet.dll	.NET functions	PHP <= 4.1.1
php_exif.dll	EXIF functions	php_mbstring.dll. And, `php_exif.dll` must be loaded *after* `php_mbstring.dll` in `php.ini`.
php_fbsql.dll	FrontBase functions	PHP <= 4.2.0
php_fdf.dll	FDF: Forms Data Format functions.	Requires: `fdftk.dll` (bundled)
php_filepro.dll	filePro functions	Read-only access
php_ftp.dll	FTP functions	Built-in since PHP 4.0.3
php_gd.dll	GD library image functions	Removed in PHP 4.3.2. Also note that truecolor functions are not available in GD1, instead, use `php_gd2.dll`.
php_gd2.dll	GD library image functions	GD2
php_gettext.dll	Gettext functions	PHP <= 4.2.0 requires `gnu_gettext.dll` (bundled), PHP >= 4.2.3 requires `libintl-1.dll`, `iconv.dll` (bundled).
php_hyperwave.dll	HyperWave functions	None

Extension	Description	Notes
php_iconv.dll	ICONV characterset conversion	Requires: `iconv-1.3.dll` (bundled), PHP >=4.2.1 `iconv.dll`
php_ifx.dll	Informix functions	Requires: Informix libraries
php_iisfunc.dll	IIS management functions	None
php_imap.dll	IMAP POP3 and NNTP functions	None
php_ingres.dll	Ingres II functions	Requires: Ingres II libraries
php_interbase.dll	InterBase functions	Requires: `gds32.dll` (bundled)
php_java.dll	Java functions	PHP <= 4.0.6 requires: `jvm.dll` (bundled)
php_ldap.dll	LDAP functions	PHP <= 4.2.0 requires `libsasl.dll` (bundled), PHP >= 4.3.0 requires `libeay32.dll`, `ssleay32.dll` (bundled)
php_mbstring.dll	Multi-Byte String functions	None
php_mcrypt.dll	Mcrypt Encryption functions	Requires: `libmcrypt.dll`
php_mhash.dll	Mhash functions	PHP >= 4.3.0 requires: `libmhash.dll` (bundled)
php_mime_magic.dll	Mimetype functions	Requires: `magic.mime` (bundled)
php_ming.dll	Ming functions for Flash	None
php_msql.dll	mSQL functions	Requires: `msql.dll` (bundled)
php_mssql.dll	MSSQL functions	Requires: `ntwdblib.dll` (bundled)
php_mysql.dll	MySQL functions	PHP >= 5.0.0, requires `libmysql.dll` (bundled)
php_mysqli.dll	MySQLi functions	PHP >= 5.0.0, requires `libmysql.dll` (`libmysqli.dll` in PHP <= 5.0.2) (bundled)
php_oci8.dll	Oracle 8 functions	Requires: Oracle 8.1+ client libraries
php_openssl.dll	OpenSSL functions	Requires: `libeay32.dll` (bundled)
php_oracle.dll	Oracle functions	Requires: Oracle 7 client libraries
php_overload.dll	Object overloading functions	Built in since PHP 4.3.0
php_pdf.dll	PDF functions	None
php_pgsql.dll	PostgreSQL functions	None
php_printer.dll	Printer functions	None
php_shmop.dll	Shared Memory functions	None
php_snmp.dll	SNMP get and walk functions	NT only!
php_soap.dll	SOAP functions	PHP >= 5.0.0
php_sockets.dll	Socket functions	None

Extension	Description	Notes
php_sybase_ct.dll	Sybase functions	Requires: Sybase client libraries
php_tidy.dll	Tidy functions	PHP >= 5.0.0
php_tokenizer.dll	Tokenizer functions	Built in since PHP 4.3.0
php_w32api.dll	W32api functions	None
php_xmlrpc.dll	XML-RPC functions	PHP >= 4.2.1 requires: `iconv.dll` (bundled)
php_xslt.dll	XSLT functions	PHP <= 4.2.0 requires `sablot.dll`, `expat.dll` (bundled). PHP >= 4.2.1 requires `sablot.dll`, `expat.dll`, `iconv.dll` (bundled).
php_yaz.dll	YAZ functions	Requires: `yaz.dll` (bundled)
php_zip.dll	Zip File functions	Read only access
php_zlib.dll	ZLib compression functions	Built in since PHP 4.3.0

Capítulo 7. Installation of PECL extensions

Introduction to PECL Installations

PHP extensions may be installed in a variety of ways. PECL is a repository of PHP extensions living within the PEAR structure, and the following demonstrates how to install these extensions.

These instructions assume */your/phpsrcdir/* is the path to the PHP source, and *extname* is the name of the PECL extension. Adjust accordingly. These instructions also assume a familiarity with the pear command.

Shared extensions may be installed by including them inside of `php.ini` using the extension PHP directive. See also the extensions_dir directive, and **dl()**. The installation methods described below do not automatically configure PHP to include these extensions, this step must be done manually.

When building PHP modules, it's important to have the appropriate versions of the required tools (autoconf, automake, libtool, etc.) See the Anonymous CVS Instructions for details on the required tools, and required versions.

Downloading PECL extensions

There are several options for downloading PECL extensions, such as:

- http://pecl.php.net

  Listed here is information like the ChangeLog, release information, requirements, revisions, etc. Although not every PECL extension has a webpage, most do.

- *pear download extname*

The pear command may also be used to download source files. Specific revisions may also be specified.

- CVS

  All PECL files reside in CVS. A web-based view may be seen at http://cvs.php.net/pecl/. To download straight from CVS, consider the following where *phpfi* is the password for user *cvsread*:

```
$ cvs -d:pserver:cvsread@cvs.php.net:/repository login
$ cvs -d:pserver:cvsread@cvs.php.net:/repository co pecl/extname
```

- Windows downloads

  Windows users may find compiled PECL binaries by downloading the *Collection of PECL modules* from the PHP Downloads page, and by retrieving a PECL Snapshot. To compile PHP under Windows, read the Win32 Build README.

PECL for Windows users

Like with any other PHP extension DLL, to install move the PECL extension DLLs into the extension_dir folder and include them within `php.ini`. For example:

```
extension=php_extname.dll
```

After doing this, restart the web server.

Compiling shared PECL extensions with PEAR

PEAR makes it easy to create shared PHP extensions. Using the pear command, do the following:

```
$ pear install extname
```

That will download the source for *extname*, and compile it on the system. This results in an `extname.so` file that may then be included in `php.ini`

In case the systems *preferred_state* is set higher than an available *extname* version, like it's set to stable and the extension is still in beta, either alter the *preferred_state* via *pear config-set* or specify a specific version of the PECL extension. For example:

```
$ pear install extname-0.1.1
```

Regardless, pear will copy this `extname.so` into the extensions directory. Adjust `php.ini` accordingly.

Compiling shared PECL extensions with phpize

If using pear is not an option, like for building shared PECL extensions from CVS, or for unreleased PECL packages, then creating a shared extension may also be done by manually using the *phpize* command. The pear command essentially does this but it may also be done manually. Assuming the source file is named `extname.tgz`, and that it was downloaded into the current directory,

consider the following:

```
$ pear download extname
$ gzip -d < extname.tgz | tar -xvf -
$ cd extname
$ phpize
$ ./configure && make
```

Upon success, this will create `extname.so` and put it into the `modules/` and/or `.libs/` directory within the `extname/` source. Move this shared extension (`extname.so`) into the PHP extensions directory, and adjust `php.ini` accordingly.

Compiling PECL extensions statically into PHP

To statically include the extension within the PHP build, put the extensions source into the `ext/` directory found in the PHP source. For example:

```
$ cd /your/phpsrcdir/ext
$ pear download extname
$ gzip -d < extname.tgz | tar -xvf -
$ mv extname-x.x.x extname
$ rm package.xml
```

This will result in the following directory:

```
/your/phpsrcdir/ext/extname
```

From here, build PHP as normal:

```
$ cd /your/phpsrcdir
$ ./buildconf
$ ./configure --help
$ ./configure --with-extname --enable-someotherext --with-foobar
$ make
$ make install
```

Whether *--enable-extname* or *--with-extname* is used depends on the extension. Typically an extension that does not require external libraries uses *--enable*. To be sure, run the following after buildconf:

```
$ ./configure --help | grep extname
```

Capítulo 8. Problemas?

Lee la FAQ

Algunos problemas son más comunes que otros. Los más comunes se pueden consultar en la FAQ de PHP en este manual.

Otros problemas

Si todavia teneis el mismo problema, alguien en la lista de correos sobre instalación de PHP, puede ayudaros. Primero, comprobar en los archivos de la lista si vuestro problema ya ha sido contestado.

Los archivos se encuentran disponibles en las página de soporte http://www.php.net/support.php. Para subscribirse a esta lista de correos mandar un correo vacío a php-install-subscribe@lists.php.net. La dirección de la lista es *php-install@lists.php.net*.

Si quereis conseguir ayuda en la lista de correo, intentar describir lo más detalladamente posible vuestro problema, los datos sobre vuestro sistema (sistema operativo que utilizais, versión de PHP, servidor web, si usais PHP como binario CGI ó como módulo, safe mode, etc...) y a ser posible código suficiente para poder reproducir vuestro problema.

Informes sobre Bugs

Si creeis que habeis encontrado un bug (error de programación) en PHP, mandarnos un informe. Probablemente los desarrolladores de PHP no lo conozcan y si no informais sobre el mismo no podrá arreglarse. Podeis informar sobre bugs a traves del sistema de seguimiento de bugs en http://bugs.php.net/. No mandar informes a la lista de correos ó en mensajes privados a los desarrolladores. El sistema de seguimiento tambien se puede utilizar para pedir nuevas caracteristicas en versiones futuras.

Lea Como informar sobre un bug antes de mandar un informe.

Capítulo 9. Configuración del comportamiento de PHP

El archivo de configuración

El archivo de configuración (llamado `php3.ini` en PHP 3, y simplemente `php.ini` a partir de PHP 4) es leído cuando arranca PHP. Para las versiones de PHP como módulo de servidor esto sólo ocurre una vez al arrancar el servidor web. Para la versión CGI y CLI, esto ocurre en cada llamada.

La localización por defecto de `php.ini` es definida en tiempo de compilación (Consultar la FAQ), pero puede ser cambiada en la versión CGI y CLI con la opción de la linea de comandos *-c*, consultar el cápitulo sobre como usar PHP desde la linea de comandos. También se puede definir la variable de entorno *PHPRC* con un "path" adicional para la búsqueda del archivo `php.ini`

Si `php-SAPI.ini` existe es usado en vez de `php.ini`.

> **Nota:** El servidor web Apache cambia al directorio raiz al arrancar, por ello PHP intentará leer el archivo `php.ini` en el directorio raiz, si existe.

Las directivas `php.ini` gestionadas por extensiones están documentadas en cada una de las páginas de las extensiones respectivamente. La lista de directivas de núcleo se encuentra disponible en el apéndice.La mayoria de directivas PHP estan listadas en **ini_set()** con los respectivos permisos y enlaces a la documentacion. Para obtener una lista completa de todas las directivas disponibles en su versión de PHP, por favor lea su archivo `php.ini`, el cual debe estar bien documentado. Alternativamente, puede encontrar útil la última versión del archivo `php.ini` desde CVS.

Ejemplo 9-1. Ejemplo `php.ini`

```
; any text on a line after an unquoted semicolon (;) is ignored
[php] ; section markers (text within square brackets) are also ignored
; Boolean values can be set to either:
;    true, on, yes
; or false, off, no, none
register_globals = off
track_errors = yes

; you can enclose strings in double-quotes
include_path = ".:/usr/local/lib/php"

; backslashes are treated the same as any other character
include_path = ".;c:\php\lib"
```

Como cambiar los valores de la configuración

Ejecución de PHP como un módulo de Apache

Cuando se usa PHP como un módulo de Apache, se pueden cambiar valores de la configuración usando directivas en los archivos de configuración de apache, `httpd.conf` y `.htaccess`. Necesitará de los privilegios "AllowOverride Options" o "AllowOverride All" para hacerlo.

Con PHP 4 y PHP 5, hay varias directivas Apache que permiten cambiar la configuración de PHP desde los archivos de configuración de apache. Para obtener una lista de que directivas son del tipo **`PHP_INI_ALL`**, **`PHP_INI_PERDIR`**, ó **`PHP_INI_SYSTEM`**, consultar la lista que se encuentra en la documentación de la función **ini_set()**.

> **Nota:** Con PHP 3, existen directivas que corresponden a cada parámetro de configuración en `php3.ini`, con el prefijo "php3_".

php_value *nombre valor*

> Asigna el valor de la directiva especificada. Puede ser usado solamente con directivas del tipo **`PHP_INI_ALL`** y **`PHP_INI_PERDIR`**. Para borrar un valor previo, asignar *none* como valor
>
> > **Nota:** No use php_value para definir valores booleanos. Debería usarse php_flag en su lugar (vea más abajo).

php_flag *nombre on|off*

> Usado para asignar una directiva de configuración booleana. Puede ser usado solamente con directivas del tipo **`PHP_INI_ALL`** y **`PHP_INI_PERDIR`**.

php_admin_value *nombre valor*

> Asigna el valor de la directiva especificada. Esto *no puede usarse* en archivos `.htaccess`. Todo tipo de directiva asignada con php_admin_value no puede ser cambiada con `.htaccess` ó directivas "virtualhost". Para borrar un valor previo, asignar *none* como valor.

php_admin_flag *nombre on|off*

> Usado para asignar una directiva de configuración booleana. Esto *no puede usarse* en archivos `.htaccess`. Todo tipo de directiva asignada con php_admin_flag no puede ser cambiada con `.htaccess` ó directivas.

Ejemplo 9-2. Ejemplo de configuración de Apache

```
<IfModule mod_php5.c>
  php_value include_path ".:/usr/local/lib/php"
  php_admin_flag safe_mode on
</IfModule>
<IfModule mod_php4.c>
  php_value include_path ".:/usr/local/lib/php"
  php_admin_flag safe_mode on
</IfModule>
<IfModule mod_php3.c>
  php3_include_path ".:/usr/local/lib/php"
  php3_safe_mode on
</IfModule>
```

Atención
Las Constantes en PHP no existen fuera de PHP. Por ejemplo, en `httpd.conf` no se pueden usar constantes PHP tales como **`E_ALL`** ó **`E_NOTICE`** para definir la directiva error_reporting, ya que no tendrá ningún significado y será evaluada como *0*. Usar los valores asociados de "bitmask" en su lugar. Estas constantes pueden ser usadas en `php.ini`

Modificación de la configuración de PHP usando el registro de Windows

Cuando se usa PHP en Windows, se pueden cambiar los valores de configuración para cada directorio por medio de los registros de Windows. Los valores de configuración se guardan en la llave de registro *HKLM\SOFTWARE\PHP\Per Directory Values*, en las subllaves correspondientes al PATH. Por ejemplo, los valores de configuracion del directorio *c:\inetpub\wwwroot* se guardarán en *HKLM\SOFTWARE\PHP\Valores Por Directorio\c\inetpub\wwwroot*. La configuración de un directorio es válida para todos los scripts ejecutados en el mismo y sus subdirectorios. Los valores en la llave deben de definirse con el nombre de la directiva de configuración de PHP y el valor tipo cadena. Las constantes PHP en los valores no son analizadas.

Otras interfaces con *PHP*

Independientemente del modo en que ejecute PHP, es posible cambiar ciertos valores en tiempo de ejecución usando **ini_set()**. Vea la documentación en la página sobre **ini_set()** para más información.

Si está interesado en una lista completa de parámetros de configuración en su sistema con sus valores actuales, puede ejecutar la función **phpinfo()**, y revisar la página resultante. También puede acceder a los valores de directivas de configuración individuales en tiempo de ejecución usando **ini_get()** o **get_cfg_var()**.

III. Referencia del lenguaje

Tabla de contenidos

Capítulo 10. Sintaxis básica

Saliendo de HTML

Para interpretar un archivo, php símplemente interpreta el texto del archivo hasta que encuentra uno de los carácteres especiales que delimitan el inicio de código PHP. El intérprete ejecuta entonces todo el código que encuentra, hasta que encuentra una etiqueta de fin de código, que le dice al intérprete que siga ignorando el código siguiente. Este mecanismo permite embeber código PHP dentro de HTML: todo lo que está fuera de las etiquetas PHP se deja tal como está, mientras que el resto se interpreta como código.

Hay cuatro conjuntos de etiquetas que pueden ser usadas para denotar bloques de código PHP. De estas cuatro, sólo 2 (<?php. . .?> y <script language="php">. . .</script>) están siempre disponibles; el resto pueden ser configuradas en el fichero de `php.ini` para ser o no aceptadas por el intérprete. Mientras que el formato corto de etiquetas (short-form tags) y el estilo ASP (ASP-style tags) pueden ser convenientes, no son portables como la versión de formato largo de etiquetas. Además, si se pretende embeber código PHP en XML o XHTML, será obligatorio el uso del formato <?php. . .?> para la compatibilidad con XML.

Las etiquetas soportadas por PHP son:

Ejemplo 10-1. Formas de escapar de HTML

```
1.  <?php echo("si quieres servir documentos XHTML o XML, haz como aqu&iacute;\n"); ?>

2.  <? echo ("esta es la m&aacute;s simple, una instrucci&oacute;n de procesado SGML \n
    <?= expression ?> Esto es una abreviatura de "<? echo expression ?>"

3.  <script language="php">
        echo ("muchos editores (como FrontPage) no
              aceptan instrucciones de procesado");
    </script>

4.  <% echo ("Opcionalmente, puedes usar las etiquetas ASP"); %>
    <%= $variable; # Esto es una abreviatura de "<% echo . . ." %>
```

El método primero, <?php. . .?>, es el más conveniente, ya que permite el uso de PHP en código XML como XHTML.

El método segundo no siempre está disponible. El formato corto de etiquetas está disponible con la función **short_tags()** (sólo PHP 3), activando el parámetro del fichero de configuración de PHP short_open_tag, o compilando PHP con la opción --enable-short-tags del comando **configure**. Aunque esté activa por defecto en php.ini-dist, se desaconseja el uso del formato de etiquetas corto.

El método cuarto sólo está disponible si se han activado las etiquetas ASP en el fichero de configuración: asp_tags.

> **Nota:** El soporte de etiquetas ASP se añadió en la versión 3.0.4.

> **Nota:** No se debe usar el formato corto de etiquetas cuando se desarrollen aplicaciones

o bibliotecas con intención de redistribuirlas, o cuando se desarrolle para servidores que no están bajo nuestro control, porque puede ser que el formato corto de etiquetas no esté soportado en el servidor. Para generar código portable y redistribuíble, asegúrate de no usar el formato corto de etiquetas.

La etiqueta de fin de bloque incluirá tras ella la siguiente línea si hay alguna presente. Además, la etiqueta de fin de bloque lleva implícito el punto y coma; no necesitas por lo tanto añadir el punto y coma final de la última línea del bloque PHP.

PHP permite estructurar bloques como:
Ejemplo 10-2. Métodos avanzados de escape

```
<?php
if ($expression) {
    ?>
    <strong>This is true.</strong>
    <?php
} else {
    ?>
    <strong>This is false.</strong>
    <?php
}
?>
```

Este ejemplo realiza lo esperado, ya que cuando PHP encuentra las etiquetas ?> de fin de bloque, empieza a escribir lo que encuentra tal cual hasta que encuentra otra etiqueta de inicio de bloque. El ejemplo anterior es, por supuesto, inventado. Para escribir bloques grandes de texto generamente es más eficiente separalos del código PHP que enviar todo el texto mediante las funciones **echo()**, **print()** o similares.

Separación de instrucciones

Las separación de instrucciones se hace de la misma manera que en C o Perl - terminando cada declaración con un punto y coma.

La etiqueta de fin de bloque (?>) implica el fin de la declaración, por lo tanto lo siguiente es ñalente:

```
<?php
    echo "This is a test";
?>

<?php echo "This is a test" ?>
```

Comentarios

PHP soporta el estilo de comentarios de 'C', 'C++' y de la interfaz de comandos de Unix. Por ejemplo:

```
<?php
    echo "This is a test"; // This is a one-line c++ style comment
    /* This is a multi line comment
       yet another line of comment */
    echo "This is yet another test";
    echo "One Final Test"; # This is shell-style style comment
?>
```

Los estilos de comentarios de una linea actualmente sólo comentan hasta el final de la linea o el

bloque actual de código PHP, lo primero que ocurra.

```
<h1>This is an <?php # echo "simple";?> example.</h1>
<p>The header above will say 'This is an example'.
```

Hay que tener cuidado con no anidar comentarios de estilo 'C', algo que puede ocurrir al comentar bloques largos de código.

```
<?php
 /*
    echo "This is a test"; /* This comment will cause a problem */
 */
?>
```

Los estilos de comentarios de una linea actualmente sólo comentan hasta el final de la linea o del bloque actual de código PHP, lo primero que ocurra. Esto implica que el código HTML tras // ?> seráa impreso: ?> sale del modo PHP, retornando al modo HTML, el comentario // no le influye.

Capítulo 11. Tipos

Introducción

PHP soporta ocho tipos primitivos.

Cuatro tipos escalares:

- **boolean**
- **integer**
- **float** (número de punto-flotante, también conocido como '**double**')
- **string**

Dos tipos compuestos:

- **array**
- **object**

Y finalmente dos tipos especiales:

- **resource**
- **NULL**

Este manual introduce también algunos pseudo-tipos por razones de legibilidad:

- mixed
- number
- callback

También puede encontrar algunas referencias al tipo "double". Considere al tipo double como el mismo que float, los dos nombres existen solo por razones históricas.

El tipo de una variable usualmente no es declarado por el programador; en cambio, es decidido en tiempo de compilación por PHP dependiendo del contexto en el que es usado la variable.

> **Nota:** Si desea chequear el tipo y valor de una cierta expresión, use **var_dump()**.

> **Nota:** Si tan solo desea una representación legible para humanos del tipo para propósitos de depuración, use **gettype()**. Para chequear por un cierto tipo, *no* use **gettype()**; en su lugar utilice las funciones *is_tipo*. Algunos ejemplos:

```
<?php
$bool = TRUE;   // un valor booleano
$str  = "foo";  // una cadena
$int  = 12;     // un entero

echo gettype($bool); // imprime "boolean"
echo gettype($str);  // imprime "string"

// Si este valor es un entere, incrementarlo en cuatro
if (is_int($int)) {
    $int += 4;
}

// Si $bool es una cadena, imprimirla
// (no imprime nada)
if (is_string($bool)) {
    echo "Cadena: $bool";
}
?>
```

Si quisiera forzar la conversión de una variable a cierto tipo, puede moldear la variable, o usar la función **settype()** sobre ella.

Note que una variable puede ser evaluada con valores diferentes en ciertas situaciones, dependiendo del tipo que posee en cada momento. Para más información, vea la sección sobre Manipulación de Tipos. Asimismo, puede encontrarse interesado en consultar las tablas de comparación de tipos, ya que éstas muestran ejemplos de las varias comparaciones relacionadas con tipos.

Booleanos

Este es el tipo más simple. Un **boolean** expresa un valor de verdad. Puede ser **`TRUE`** or **`FALSE`**.

> **Nota:** El tipo booleano fue introducido en PHP 4.

Sintaxis

Para especificar un literal booleano, use alguna de las palabras clave **`TRUE`** o **`FALSE`**. Ambas son insensibles a mayúsculas y minúsculas.

```
<?php
$foo = True; // asignar el valor TRUE a $foo
?>
```

Usualmente se usa algún tipo de operador que deuelve un valor **boolean**, y luego éste es pasado a

una estructura de control.

```
<?php
// == es un operador que prueba por
// igualdad y devuelve un booleano
if ($accion == "mostrar_version") {
    echo "La versi&oacute;n es 1.23";
}

// esto no es necesario...
if ($mostrar_separadores == TRUE) {
    echo "<hr>\n";
}

// ...porque se puede escribir simplemente
if ($mostrar_separadores) {
    echo "<hr>\n";
}
?>
```

Conversión a booleano

Para convertir explícitamente un valor a **boolean**, use el moldeamiento *(bool)* o *(boolean)*. Sin embargo, en la mayoría de casos no es necesario usar el moldeamiento, ya que un valor será convertido automáticamente si un operador, función o estructura de control requiere un argumento tipo **boolean**.

Vea también Manipulación de Tipos.

Cuando se realizan conversiones a **boolean**, los siguientes valores son considerados **`FALSE`**:

- el boolean **`FALSE`** mismo
- el integer 0 (cero)
- el float 0.0 (cero)
- el valor string vacío, y el string "0"
- un array con cero elementos
- un object con cero variables miembro
- el tipo especial NULL (incluyendo variables no definidas)

Cualquier otro valor es considerado **`TRUE`** (incluyendo cualquier resource).

Aviso
Ã‚Â¡*-1* es considerado **`TRUE`**, como cualquier otro número diferente a cero (ya sea negativo o positivo)!

```php
<?php
var_dump((bool) "");        // bool(false)
var_dump((bool) 1);         // bool(true)
var_dump((bool) -2);        // bool(true)
var_dump((bool) "foo");     // bool(true)
var_dump((bool) 2.3e5);     // bool(true)
var_dump((bool) array(12)); // bool(true)
var_dump((bool) array());   // bool(false)
var_dump((bool) "false");   // bool(true)
?>
```

Enteros

Un **integer** es un número del conjunto Z = {..., -2, -1, 0, 1, 2, ...}.

Vea también: Entero de longitud arbitraria / GMP, Números de punto flotante, y Precisión arbitraria / BCMath

---

Sintaxis

Los enteros pueden ser especificados en notación decimal (base-10), hexadecimal (base-16) u octal (base-8), opcionalmente precedidos por un signo (- o +).

Si usa la notación octal, debe preceder el número con un *0* (cero), para usar la notación hexadecimal, preceda el número con *0x*.

Ejemplo 11-1. Literales tipo entero

```php
<?php
$a = 1234; // numero decimal
$a = -123; // un numero negativo
$a = 0123; // numero octal (ñalente al 83 decimal)
$a = 0x1A; // numero hexadecimal (ñalente al 26 decimal)
?>
```

Formalmente, la posible estructura para literales enteros es:

```
decimal     : [1-9][0-9]*
            | 0

hexadecimal : 0[xX][0-9a-fA-F]+

octal       : 0[0-7]+

integer     : [+-]?decimal
            | [+-]?hexadecimal
            | [+-]?octal
```

El tamaño de un entero es dependiente de la plataforma, aunque un valor máximo de aproximadamente dos billones es el valor usual (lo que es un valor de 32 bits con signo). PHP no soporta enteros sin signo.

Aviso
Si un dígito inválido es pasado a un entero octal (p.ej. 8 o 9), el resto del número es ignorado.

Ejemplo 11-2. Curiosidad de valores octales

```php
<?php
var_dump(01090); // 010 octal = 8 decimal
?>
```

Desbordamiento de enteros

Si especifica un número más allá de los límites del tipo **integer**, será interpretado en su lugar como un **float**. Asimismo, si realiza una operación que resulta en un número más allá de los límites del tipo **integer**, un **float** es retornado en su lugar.

```
<?php
$numero_grande =  2147483647;
var_dump($numero_grande);
// output: int(2147483647)

$numero_grande =  2147483648;
var_dump($numero_grande);
// output: float(2147483648)

// esto tambien ocurre con los enteros indicados como hexadecimales:
var_dump( 0x80000000 );
// output: float(2147483648)

$millon = 1000000;
$numero_grande =  50000 * $millon;
var_dump($numero_grande);
// output: float(50000000000)
?>
```

Aviso
Desafortunadamente, había un fallo en PHP que provocaba que esto no siempre funcionara correctamente cuando se presentaban números negativos. Por ejemplo: cuando hace *-50000 * $millon*, el resultado será *-429496728*. Sin embargo, cuando ambos operandos son positivos no se presenta ningún problema. Este problema fue resuelto en PHP 4.1.0.

No hay un operador de división de enteros en PHP. *1/2* produce el **float** *0.5*. Puede moldear el valor a un entero para asegurarse de redondearlo hacia abajo, o puede usar la función **round()**.

```
<?php
var_dump(25/7);         // float(3.5714285714286)
var_dump((int) (25/7)); // int(3)
var_dump(round(25/7));  // float(4)
?>
```

Conversión a entero

Para convertir explícitamente un valor a **integer**, use alguno de los moldeamientos *(int)* o *(integer)*. Sin embargo, en la mayoría de casos no necesita usar el moldeamiento, ya que un valor será convertido automáticamente si un operador, función o estructura de control requiere un argumento tipo **integer**. También puede convertir un valor a entero con la función **intval()**.

Vea también Manipulación de Tipos.

Desde booleans

FALSE producirá *0* (cero), y **TRUE** producirá *1* (uno).

Desde números de punto flotante

Cuando se realizan conversiones desde un flotante a un entero, el número será redondeado *hacia cero*.

Si el flotante se encuentra más allá de los límites del entero (usualmente *+/- 2.15e+9 = 2^31*), el resultado es indefinido, ya que el flotante no tiene suficiente precisión para dar un resultado entero exacto. No se producirá una advertencia, Ã‚Â¡ni siquiera una noticia en este caso!

Aviso

Nunca moldee una fracción desconocida a **integer**, ya que esto en ocasiones produce resultados inesperados.
`<?php`
`echo (int) ( (0.1+0.7) * 10 ); // imprime 7!`
`?>`
Para más información, consulte la advertencia sobre precisión-flotante.

Desde cadenas

Vea Conversión de cadenas a números

Desde otros tipos

Atención
El comportamiento de convertir desde entero no es definido para otros tipos. Actualmente, el comportamiento es el mismo que si el valor fuera antes convertido a booleano. Sin embargo, *no* confíe en este comportamiente, ya que puede ser modificado sin aviso.

Números de punto flotante

Los números de punto flotante (también conocidos como "flotantes", "dobles" o "números reales") pueden ser especificados usando cualquiera de las siguientes sintaxis:

```
<?php
$a = 1.234;
$b = 1.2e3;
$c = 7E-10;
?>
```

Formalmente:

```
LNUM          [0-9]+
DNUM          ([0-9]*[\.]{LNUM}) | ({LNUM}[\.][0-9]*)
EXPONENT_DNUM ( ({LNUM} | {DNUM}) [eE][+-]? {LNUM})
```

El tamaño de un flotante depende de la plataforma, aunque un valor común consiste en un máximo de ~1.8e308 con una precisión de aproximadamente 14 dígitos decimales (lo que es un valor de 64 bits en formato IEEE).

Precisión del punto flotante
Es bastante común que algunas fracciones decimales simples como *0.1* o *0.7* no puedan ser convertidas a su representación binaria interna sin perder un poco de precisión. Esto puede llevar a resultados confusos: por ejemplo, *floor((0.1+0.7)*10)* usualmente devolverá *7* en lugar del esperado *8* ya que el resultado de la representación interna es en realidad algo como *7.9999999999...*. Esto se encuentra relacionado al hecho de que es imposible expresar de forma exacta algunas fracciones en notación decimal con un número finito de dígitos. Por ejemplo, *1/3* en forma decimal se convierte en *0.3333333. . ..* Así que nunca confíe en resultados de números flotantes hasta el último dígito, y nunca compare números de punto flotante para conocer si son ñalentes. Si realmente necesita una mejor precisión, es buena idea que use las funciones matemáticas de precisión arbitraria o las funciones gmp en su lugar.

Conversión a flotante

Para más información sobre cuándo y cómo son convertidas las cadenas a flotantes, vea la sección titulada Conversión de cadenas a números. Para valores de otros tipos, la conversión es la misma que si el valor hubiese sido convertido a entero y luego a flotante. Vea la sección Conversión a entero para más información.

Cadenas

Un valor **string** es una serie de caracteres. En PHP, un caracter es lo mismo que un byte, es decir, hay exactamente 256 tipos de caracteres diferentes. Esto implica también que PHP no tiene soporte nativo de Unicode. Vea **utf8_encode()** y **utf8_decode()** para conocer sobre el soporte Unicode.

> **Nota:** El que una cadena se haga muy grande no es un problema. PHP no impone límite práctico alguno sobre el tamaño de las cadenas, así que no hay ninguna razón para preocuparse sobre las cadenas largas.

Sintaxis

Un literal de cadena puede especificarse en tres formas diferentes.

- comillas simples
- comillas dobles
- sintaxis heredoc

Comillas simples

La forma más simple de especificar una cadena sencilla es rodearla de comillas simples (el caracter ').

Para especificar una comilla sencilla literal, necesita escaparla con una barra invertida (\), como en muchos otros lenguajes. Si una barra invertida necesita aparecer antes de una comilla sencilla o al final de la cadena, necesitará doblarla. Note que si intenta escapar cualquier otro caracter, ¡la barra invertida será impresa también! De modo que, por lo general, no hay necesidad de escapar la barra invertida misma.

> **Nota:** En PHP 3, se generará una advertencia de nivel *E_NOTICE* cuando esto ocurra.

> **Nota:** A diferencia de las otras dos sintaxis, las variables y secuencias de escape para caracteres especiales *no* serán expandidas cuando ocurren al interior de cadenas entre comillas sencillas.

```
<?php
echo 'esta es una cadena simple';

echo 'Tambi&eacute;n puede tener saltos de l&iacute;nea embebidos
en las cadenas de esta forma, ya que
es v&aacute;lido';

// Imprime: Arnold dijo una vez: "I'll be back"
echo 'Arnold dijo una vez: "I\'ll be back"';

// Imprime: Ha eliminado C:\*.*?
echo 'Ha eliminado C:\\*.*?';

// Imprime: Ha eliminado C:\*.*?
echo 'Ha eliminado C:\*.*?';

// Imprime: Esto no va a expandirse: \n una nueva linea
echo 'Esto no va a expandirse: \n una nueva linea';

// Imprime: Las variables no se $expanden $tampoco
echo 'Las variables no se $expanden $tampoco';
?>
```

Comillas dobles

Si la cadena se encuentra rodeada de comillas dobles ("), PHP entiende más secuencias de escape para caracteres especiales:

Tabla 11-1. Caracteres escapados

secuencia	significado
\n	alimentación de línea (LF o 0x0A (10) en ASCII)
\r	retorno de carro (CR o 0x0D (13) en ASCII)
\t	tabulación horizontal (HT o 0x09 (9) en ASCII)
\\	barra invertida
\$	signo de dólar
\"	comilla-doble
\[0-7]{1,3}	la secuencia de caracteres que coincide con la expresión regular es un caracter en notación octal
\x[0-9A-Fa-f]{1,2}	la secuencia de caracteres que coincide con la expresión regular es un caracter en notación hexadecimal

Nuevamente, si intenta escapar cualquier otro caracter, ¡la barra invertida será impresa también!

Pero la característica más importante de las cadenas entre comillas dobles es el hecho de que los nombres de variables serán expandidos. Vea procesamiento de cadenas para más detalles.

Heredoc

Otra forma de delimitar cadenas es mediante el uso de la sintaxis heredoc ("<<<"). Debe indicarse un identificador después de la secuencia <<<, luego la cadena, y luego el mismo identificador para cerrar la cita.

El identificador de cierre *debe* comenzar en la primera columna de la línea. Asimismo, el identificador usado debe seguir las mismas reglas que cualquier otra etiqueta en PHP: debe contener solo caracteres alfanuméricos y de subrayado, y debe iniciar con un caracter no-dígito o de subrayado.

Aviso
Es muy importante notar que la línea con el identificador de cierre no contenga otros caracteres, excepto *quizás* por un punto-y-coma (*;*). Esto quiere decir en especial que el identificador *no debe usar sangría*, y no debe haber espacios o tabuladores antes o después del punto-y-coma. Es importante también notar que el primer caracter antes del identificador de cierre debe ser un salto de línea, tal y como lo defina su sistema operativo. Esto quiere decir *\r* en Macintosh, por ejemplo. Si esta regla es rota y el identificador de cierre no es "limpio", entonces no se considera un identificador de cierre y PHP continuará en busca de uno. Si, en tal caso, no se encuentra un identificador de cierre apropiado, entonces un error del analizador sintáctico resultará con el número de línea apuntando al final del script.

El texto heredoc se comporta tal como una cadena entre comillas dobles, sin las comillas dobles. Esto quiere decir que no necesita escapar tales comillas en sus bloques heredoc, pero aun puede usar los códigos de escape listados anteriormente. Las variables son expandidas, aunque debe tenerse el mismo cuidado cuando se expresen variables complejas al interior de un segmento heredoc, al igual que con otras cadenas.

Ejemplo 11-3. Ejemplo de uso de una cadena heredoc

```
<?php
$cadena = <<<FIN
Ejemplo de una cadena
que se extiende por varias l&iacute;neas
usando la sintaxis heredoc.
FIN;

/* Un ejemplo mas complejo, con variables. */
class foo
{
    var $foo;
    var $bar;

    function foo()
    {
        $this->foo = 'Foo';
        $this->bar = array('Bar1', 'Bar2', 'Bar3');
    }
}

$foo = new foo();
$nombre = 'MiNombre';

echo <<<FIN
Mi nombre es "$nombre". Estoy imprimiendo algo de $foo->foo.
Ahora, estoy imprimiendo algo de {$foo->bar[1]}.
Esto deber&iacute;a imprimir una letra 'A' may&uacute;scula: \x41
FIN;
?>
```

Nota: El soporte heredoc fue agregado en PHP 4.

Procesamiento de variables

Cuando una cadena es especificada en comillas dobles o al interior de un bloque heredoc, las variables son interpretadas en su interior.

Existen dos tipos de sintaxis: una simple y una compleja. La sintaxis simple es la más común y conveniente. Esta ofrece una forma de interpretar una variable, un valor **array**, o una propiedad de un **object**.

La sintaxis compleja fue introducida en PHP 4, y puede reconocerse por las llaves que rodean la expresión.

Sintaxis simple

Si un signo de dólar (*$*) es encontrado, el analizador sintáctico tomará ambiciosamente tantos lexemas como le sea posible para formar un nombre de variable válido. Rodee el nombre de la variable de llaves si desea especificar explícitamente el final del nombre.

```
<?php
$cerveza = 'Heineken';
echo "El sabor de varias $cerveza's es excelente"; // funciona, "'" no es un caracter v
echo "Tom&oacute; algunas $cervezas";   // no funciona, 's' es un caracter valido para
echo "Tom&oacute; algunas ${cerveza}s"; // funciona
echo "Tom&oacute; algunas {$cerveza}s"; // funciona
?>
```

De forma similar, puede hacer que un índice de un **array** o una propiedad de un **object** sean interpretados. En el caso de los índices de matrices, el corchete cuadrado de cierre (*]*) marca el final

del índice. Para las propiedades de objetos, se aplican las mismas reglas de las variables simples, aunque con las propiedades de objetos no existe un truco como el que existe con las variables.

```
<?php
// Estos ejemplos son especificos al uso de matrices al interior de
// cadenas. Cuando se encuentre por fuera de una cadena, siempre rodee
// de comillas las claves tipo cadena de su matriz, y no use
// {llaves} por fuera de cadenas tampoco.

// Mostremos todos los errores
error_reporting(E_ALL);

$frutas = array('fresa' => 'roja', 'banano' => 'amarillo');

// Funciona pero note que esto trabaja de forma diferente por fuera de
// cadenas entre comillas
echo "Un banano es $frutas[banano].";

// Funciona
echo "Un banano es {$frutas['banano']}.";

// Funciona, pero PHP musca una constante llamada banano primero, como
// se describe mas adelante.
echo "Un banano es {$frutas[banano]}.";

// No funciona, use llaves. Esto resulta en un error de analisis sintactico.
echo "Un banano es $frutas['banano'].";

// Funciona
echo "Un banano es " . $frutas['banano'] . ".";

// Funciona
echo "Este cuadro tiene $cuadro->ancho metros de ancho.";

// No funciona. Para una solucion, vea la sintaxis compleja.
echo "Este cuadro tiene $cuadro->ancho00 cent&iacute;metros de ancho.";
?>
```

Para cualquier cosa más sofisticada, debería usarse la sintaxis compleja.

Sintaxis compleja (llaves)

Esta no es llamada compleja porque la sintaxis sea compleja, sino porque es posible incluir expresiones complejas de esta forma.

De hecho, de esta forma puede incluir cualquier valor que sea parte del espacio de nombres al interior de cadenas. Simplemente escriba la expresión en la misma forma que lo haría si se encontrara por fuera de una cadena, y luego la ubica entre { y }. Ya que no es posible escapar '{', esta sintaxis será reconocida únicamente cuando el caracter $ se encuentra inmediatamente después de {. (Use "{\$" o "\{$" para obtener una secuencia literal "{$"). Algunos ejemplos para aclarar el asunto:

```
<?php
// Mostremos todos los errores
error_reporting(E_ALL);

$genial = 'fant&aacute;stico';

// No funciona, imprime: Esto es { fant&aacute;stico}
echo "Esto es { $genial}";

// Funciona, imprime: Esto es fant&aacute;stico
echo "Esto es {$genial}";
echo "Esto es ${genial}";

// Funciona
echo "Este cuadro tiene {$cuadro->ancho}00 cent&iacute;metros de ancho.";

// Funciona
echo "Esto funciona: {$matriz[4][3]}";

// Esto esta mal por la misma razon por la que $foo[bar] esta mal por
// fuera de una cadena. En otras palabras, aun funciona pero ya que
// PHP busca primero una constante llamada foo, genera un error de
// nivel E_NOTICE (constante indefinida).
echo "Esto esta mal: {$matriz[foo][3]}";

// Funciona. Cuando se usan matrices multi-dimensionales, use siempre
// llaves alrededor de las matrices al interior de cadenas
echo "Esto funciona: {$matriz['foo'][3]}";

// Funciona.
echo "Esto funciona: " . $arr['foo'][3];

echo "Puede incluso escribir {$obj->valores[3]->nombre}";

echo "Este es el valor de la variable llamada $nombre: {${$nombre}}";
?>
```

Acceso a cadenas y modificación por caracter

Los caracteres al interior de una cadena pueden ser consultados y modificados al especificar el desplazamiento, comenzando en cero, del caracter deseado después de la cadena entre llaves.

> **Nota:** Para efectos de compatibilidad con versiones anteriores, aun puede usar corchetes tipo matriz para el mismo propósito. Sin embargo, esta sintaxis es obsoleta a partir de PHP 4.

Ejemplo 11-4. Algunos ejemplos de cadenas

```
<?php
// Obtener el primer caracter de una cadena
$cadena = 'Esta es una prueba.';
$primer = $cadena{0};

// Obtener el tercer caracter de una cadena
$tercer = $cadena{2};

// Obtener el ultimo caracter de una cadena.
$cadena = 'Esta es tambien una prueba.';
$ultimo = $cadena{strlen($cadena)-1};

// Modificar el ultimo caracter de una cadena
$cadena = 'Observe el mar';
$cadena{strlen($cadena)-1} = 'l';

?>
```

Funciones y operadores útiles

Las cadenas pueden ser concatenadas usando el operador '.' (punto). Note que el operador '+' (adición) no funciona para este propósito. Por favor refiérase a la sección Operadores de cadena para más información.

Existen bastantes funciones útiles para la modificación de cadenas.

Vea la sección de funciones de cadena para consultar funciones de uso general, o las funciones de expresiones regulares para búsquedas y reemplazos avanzados (en dos sabores: Perl y POSIX extendido).

Existen también funciones para cadenas tipo URL, y funciones para encriptar/descifrar cadenas (mcrypt y mhash).

Finalmente, si aun no ha encontrado lo que busca, vea también las funciones de tipo de caracter.

Conversión a cadena

Es posible convertir un valor a una cadena usando el moldeamiento *(string)*, o la función **strval()**. La conversión a cadena se realiza automáticamente para usted en el contexto de una expresión cuando se necesita una cadena. Esto ocurre cuando usa las funciones **echo()** o **print()**, o cuando compara el valor de una variable con una cadena. El contenido de las secciones del manual sobre Tipos y Manipulación de Tipos ayudan a aclarar este hecho. Vea también **settype()**.

Un valor **boolean** **TRUE** es convertido a la cadena *"1"*, el valor **FALSE** se representa como *""* (una cadena vacía). De esta forma, usted puede convertir de ida y vuelta entre valores booleanos y de cadena.

Un número **integer** o de punto flotante (**float**) es convertido a una cadena que representa el número con sus dígitos (incluyendo la parte del exponente para los números de punto flotante).

Las matrices son siempre convertidas a la cadena *"Array"*, de modo que no puede volcar los contenidos de un valor **array** con **echo()** o **print()** para ver lo que se encuentra en su interior. Para ver un elemento, usted tendría que hacer algo como *echo $arr['foo']*. Vea más adelante algunos consejos sobre el volcado/vista del contenido completo.

Los objetos son convertidos siempre a la cadena *"Object"*. Si quisiera imprimir los valores de variables miembro de un **object** para efectos de depuración, lea los parágrafos siguientes. Si quiere conocer el nombre de clase del cual un objeto dado es instancia, use **get_class()**.

Los recursos son siempre convertidos a cadenas con la estructura *"Resource id #1"* en donde *1* es el número único del valor **resource** asignado por PHP durante tiempo de ejecución. Si quisiera obtener el tipo del recurso, use **get_resource_type()**.

NULL se convierte siempre a una cadena vacía.

Como puede apreciar, el imprimir matrices, objetos o recursos no le ofroce información útil sobre los valores mismos. Consulte las funciones **print_r()** y **var_dump()** para conocer mejores formas de imprimir valores para depuración.

También puede convertir valores PHP a cadenas y almacenarlas permanentemente. Este método es

conocido como seriación, y puede ser efectuado con la función **serialize()**. También puede seriar valores PHP a estructuras XML, si cuenta con soporte WDDX en su configuración de PHP.

Conversión de cadenas a números

Cuando una cadena es evaluada como un valor numérico, el valor resultante y su tipo son determinados como sigue.

La cadena será evaluada como un **float** si contiene cualquier caracter entre '.', 'e', o 'E'. De otra forma, evaluará como un entero.

El valor es dado por la porción inicial de la cadena. Si la cadena comienza con datos numéricos válidos, éstos serán el valor usado. De lo contrario, el valor será 0 (cero). Un signo opcional es considerado un dato numérico válido, seguido por uno o más dígitos (que pueden contener un punto decimal), seguidos por un exponente opcional. El exponente es una 'e' o 'E' seguida de uno o más dígitos.

```
<?php
$foo = 1 + "10.5";                // $foo es flotante (11.5)
$foo = 1 + "-1.3e3";              // $foo es flotante (-1299)
$foo = 1 + "bob-1.3e3";           // $foo es entero (1)
$foo = 1 + "bob3";                // $foo es entero (1)
$foo = 1 + "10 Cerditos";         // $foo es entero (11)
$foo = 4 + "10.2 Cerditos";       // $foo es flotante (14.2)
$foo = "10.0 cerdos " + 1;        // $foo es flotante (11)
$foo = "10.0 cerdos " + 1.0;      // $foo es flotante (11)
?>
```

Para más información sobre esta conversión, vea la página del manual Unix sobre strtod(3).

Si quisiera probar cualquiera de los ejemplos presentados en esta sección, puede cortar y pegar los ejemplos e insertar la siguiente línea para verificar por sí mismo lo que está sucediendo:

```
<?php
echo "\$foo==$foo; tipo es " . gettype ($foo) . "<br />\n";
?>
```

No espere obtener el código de un caractar convirtiéndolo a un entero (como lo haría en C, por ejemplo). Use las funciones **ord()** y **chr()** para convertir entre códigos de caracter y caracteres.

Matrices

Una matriz en PHP es en realidad un mapa ordenado. Un mapa es un tipo de datos que asocia *valores* con *claves*. Este tipo es optimizado en varias formas, de modo que puede usarlo como una matriz real, o una lista (vector), tabla asociativa (caso particular de implementación de un mapa), diccionario, colección, pila, cola y probablemente más. Ya que puede tener otra matriz PHP como valor, es realmente fácil simular árboles.

Una explicación sobre tales estructuras de datos se encuentra por fuera del propósito de este manual, pero encontrará al menos un ejemplo de cada uno de ellos. Para más información, le referimos a literatura externa sobre este amplio tema.

Sintaxis

Especificación con array()

Un **array** puede ser creado por la construcción de lenguaje **array()**. Ésta toma un cierto número de parejas *clave* => *valor* separadas con coma.

```
array( [clave =>] valor
     , ...
     )
// clave puede ser un integer o string
// valor puede ser cualquier valor
```

```
<?php
$matriz = array("foo" => "bar", 12 => true);

echo $matriz["foo"]; // bar
echo $matriz[12];    // 1
?>
```

Una *clave* puede ser un *integer* o un **string**. Si una clave es la representación estándar de un **integer**, será interpretada como tal (es decir, *"8"* será interpretado como *8*, mientras que *"08"* será interpretado como *"08"*). No existen tipos diferentes para matrices indexadas y asociativas en PHP; sólo existe un tipo de matriz, el cual puede contener índices tipo entero o cadena.

Un valor puede ser de cualquier tipo en PHP.

```
<?php
$matriz = array("unamatriz" => array(6 => 5, 13 => 9, "a" => 42));

echo $matriz["unamatriz"][6];     // 5
echo $matriz["unamatriz"][13];    // 9
echo $matriz["unamatriz"]["a"];   // 42
?>
```

Si no especifica una clave para un valor dado, entonces es usado el máximo de los índices enteros, y la nueva clave será ese valor máximo + 1. Si especifica una clave que ya tiene un valor asignado, ése valor será sobrescrito.

```
<?php
// Esta matriz es la misma que ...
array(5 => 43, 32, 56, "b" => 12);

// ...esta matriz
array(5 => 43, 6 => 32, 7 => 56, "b" => 12);
?>
```

Aviso
A partir de PHP 4.3.0, el comportamiento de generación de índices descrito ha cambiado. Ahora, si agrega un elemento a una matriz cuya clave máxima actual es un valor negativo, entonces la siguiente clave creada será cero (*0*). Anteriormente, el nuevo índice hubiera sido establecido a la clave mayor existente + 1, al igual que con los índices positivos.

Al usar **TRUE** como clave, el valor será evaluado al **integer** *1*. Al usar **FALSE** como clave, el valor será evaluado al **integer** *0*. Al usar *NULL* como clave, el valor será evaluado a una cadena vacía. El uso de una cadena vacía como clave creará (o reemplazará) una clave con la cadena vacía y su valor; no es lo mismo que usar corchetes vacíos.

No es posible usar matrices u objetos como claves. Al hacerlo se producirá una advertencia: *Illegal offset type*.

Creación/modificación con sintaxis de corchetes cuadrados

Es posible modificar una matriz existente al definir valores explícitamente en ella.

Esto es posible al asignar valores a la matriz al mismo tiempo que se especifica la clave entre corchetes. También es posible omitir la clave, agregar una pareja vacía de corchetes ("*[]*") al nombre de la variable en ese caso.

```
$matriz[clave] = valor;
$matriz[] = valor;
// clave puede ser un integer o string
// valor puede ser cualquier valor
```

Si *$matriz* no existe aun, ésta será creada. De modo que esta es también una forma alternativa de especificar una matriz. Para modificar un cierto valor, simplemente asigne un nuevo valor a un elemento especificado con su clave. Si desea remover una pareja clave/valor, necesita eliminarla mediante **unset()**.

```
<?php
$matriz = array(5 => 1, 12 => 2);

$matriz[] = 56;    // Esto es igual que $matriz[13] = 56;
                   // en este punto del script

$matriz["x"] = 42; // Esto agrega un nuevo elemento a la
                   // matriz con la clave "x"

unset($matriz[5]); // Esto elimina el elemento de la matriz

unset($matriz);    // Esto elimina la matriz completa
?>
```

Nota: Como se menciona anteriormente, si provee los corchetes sin ninguna clave especificada, entonces se toma el máximo de los índices enteros existentes, y la nueva clave será ese valor máximo + 1. Si no existen índices enteros aun, la clave será *0* (cero). Si especifica una clave que ya tenía un valor asignado, el valor será reemplazado.

Aviso
A partir de PHP 4.3.0, el comportamiento de generación de índices descrito ha cambiado. Ahora, si agrega un elemento al final de una matriz en la que la clave máxima actual es negativa, la siguiente clave creada será cero (*0*). Anteriormente, el nuevo índice hubiera sido definido como la mayor clave + 1, al igual que ocurre con los índices positivos.

Note que la clave entera máxima usada para este caso *no necesita existir actualmente en la matriz*. Tan solo debe haber existido en la matriz en algún punto desde que la matriz haya sido re-indexada. El siguiente ejemplo ilustra este caso:

```php
<?php
// Crear una matriz simple.
$matriz = array(1, 2, 3, 4, 5);
print_r($matriz);

// Ahora eliminar cada item, pero dejar la matriz misma intacta:
foreach ($matriz as $i => $valor) {
    unset($matriz[$i]);
}
print_r($matriz);

// Agregar un item (note que la nueva clave es 5, en lugar de 0 como
// podria esperarse).
$matriz[] = 6;
print_r($matriz);

// Re-indexar:
$matriz = array_values($matriz);
$matriz[] = 7;
print_r($matriz);
?>
```

El anterior ejemplo produciría la siguiente salida:

```
Array
(
    [0] => 1
    [1] => 2
    [2] => 3
    [3] => 4
    [4] => 5
)
Array
(
)
Array
(
    [5] => 6
)
Array
(
    [0] => 6
    [1] => 7
)
```

Funciones útiles

Existe un buen número de funciones útiles para trabajar con matrices. Consulte la sección funciones de matrices.

> **Nota:** La función **unset()** le permite remover la definición de claves de una matriz. Tenga en cuenta que la matriz NO es re-indexada. Si sólo usa "índices enteros comunes" (comenzando desde cero, incrementando en uno), puede conseguir el efecto de re-indexación usando **array_values()**.

```
<?php
$a = array(1 => 'uno', 2 => 'dos', 3 => 'tres');
unset($a[2]);
/* producira una matriz que hubiera sido definida como
   $a = array(1 => 'uno', 3 => 'tres');
   y NO
   $a = array(1 => 'uno', 2 =>'tres');
*/

$b = array_values($a);
// Ahora $b es array(0 => 'uno', 1 =>'tres')
?>
```

La estructura de control foreach existe específicamente para las matrices. Ésta provee una manera fácil de recorrer una matriz.

Recomendaciones sobre matrices y cosas a evitar

¿Porqué es incorrecto *$foo[bar]*?

Siempre deben usarse comillas alrededor de un índice de matriz tipo cadena literal. Por ejemplo, use $foo['bar'] y no $foo[bar]. ¿Pero qué está mal en $foo[bar]? Es posible que haya visto la siguiente sintaxis en scripts viejos:

```
<?php
$foo[bar] = 'enemigo';
echo $foo[bar];
// etc
?>
```

Esto está mal, pero funciona. Entonces, ¿porqué está mal? La razón es que este código tiene una constante indefinida (bar) en lugar de una cadena ('bar' - note las comillas), y puede que en el futuro PHP defina constantes que, desafortunadamente para su código, tengan el mismo nombre. Funciona porque PHP automáticamente convierte una *cadena pura* (una cadena sin comillas que no corresponda con símbolo conocido alguno) en una cadena que contiene la cadena pura. Por ejemplo, si no se ha definido una constante llamada **`bar`**, entonces PHP reemplazará su valor por la cadena *'bar'* y usará ésta última.

> **Nota:** Esto no quiere decir que *siempre* haya que usar comillas en la clave. No querrá usar comillas con claves que sean constantes o variables, ya que en tal caso PHP no podrá interpretar sus valores.

```
<?php
error_reporting(E_ALL);
ini_set('display_errors', true);
ini_set('html_errors', false);
// Matriz simple:
$matriz = array(1, 2);
$conteo = count($matriz);
for ($i = 0; $i < $conteo; $i++) {
    echo "\nRevisando $i: \n";
    echo "Mal: " . $matriz['$i'] . "\n";
    echo "Bien: " . $matriz[$i] . "\n";
    echo "Mal: {$matriz['$i']}\n";
    echo "Bien: {$matriz[$i]}\n";
}
?>
```

Nota: La salida del anterior fragmento es:

```
Revisando 0:
Notice: Undefined index:  $i in /path/to/script.html on line 9
Mal:
Bien: 1
Notice: Undefined index:  $i in /path/to/script.html on line 11
Mal:
Bien: 1

Revisando 1:
Notice: Undefined index:  $i in /path/to/script.html on line 9
Mal:
Bien: 2
Notice: Undefined index:  $i in /path/to/script.html on line 11
Mal:
Bien: 2
```

Más ejemplos para demostrar este hecho:

```
<?php
// Mostrar todos los errores
error_reporting(E_ALL);

$matriz = array('fruta' => 'manzana', 'vegetal' => 'zanahoria');

// Correcto
print $matriz['fruta'];   // manzana
print $matriz['vegetal']; // zanahoria

// Incorrecto. Esto funciona pero tambi&eacute;n genera un error de PHP de
// nivel E_NOTICE ya que no hay definida una constante llamada fruta
//
// Notice: Use of undefined constant fruta - assumed 'fruta' in...
print $matriz[fruta];    // manzana

// Definamos una constante para demostrar lo que pasa. Asignaremos el
// valor 'vegetal' a una constante llamada fruta.
define('fruta', 'vegetal');

// Note la diferencia ahora
print $matriz['fruta'];  // manzana
print $matriz[fruta];    // zanahoria

// Lo siguiente esta bien ya que se encuentra al interior de una
// cadena. Las constantes no son procesadas al interior de
// cadenas, asi que no se produce un error E_NOTICE aqui
print "Hola $matriz[fruta]";  // Hola manzana

// Con una excepcion, los corchetes que rodean las matrices al
// interior de cadenas permiten el uso de constantes
print "Hola {$matriz[fruta]}";    // Hola zanahoria
print "Hola {$matriz['fruta']}";  // Hola manzana

// Esto no funciona, resulta en un error de interprete como:
// Parse error: parse error, expecting T_STRING' or T_VARIABLE' or T_NUM_STRING'
// Esto se aplica tambien al uso de autoglobales en cadenas, por supuesto
print "Hola $matriz['fruta']";
print "Hola $_GET['foo']";

// La concatenacion es otra opcion
print "Hola " . $matriz['fruta']; // Hola manzana
?>
```

Cuando habilita **error_reporting()** para mostrar errores de nivel **E_NOTICE** (como por ejemplo definiendo el valor **E_ALL**) verá estos errores. Por defecto, error_reporting se encuentra configurado para no mostrarlos.

Tal y como se indica en la sección de sintaxis, debe existir una expresión entre los corchetes cuadrados ('[' y ']'). Eso quiere decir que puede escribir cosas como esta:

```
<?php
echo $matriz[alguna_funcion($bar)];
?>
```

Este es un ejemplo del uso de un valor devuelto por una función como índice de matriz. PHP también conoce las constantes, tal y como ha podido apreciar aquellas *E_** antes.

```
<?php
$descripciones_de_error[E_ERROR]   = "Un error fatal ha ocurrido";
$descripciones_de_error[E_WARNING] = "PHP produjo una advertencia";
$descripciones_de_error[E_NOTICE]  = "Esta es una noticia informal";
?>
```

Note que *E_ERROR* es también un identificador válido, asi como *bar* en el primer ejemplo. Pero el último ejemplo es ñalente a escribir:

```
<?php
$descripciones_de_error[1] = "Un error fatal ha ocurrido";
$descripciones_de_error[2] = "PHP produjo una advertencia";
$descripciones_de_error[8] = "Esta es una noticia informal";
?>
```

ya que *E_ERROR* es igual a *1*, etc.

Tal y como lo hemos explicado en los anteriores ejemplos, *$foo[bar]* aun funciona pero está mal. Funciona, porque debido a su sintaxis, se espera que *bar* sea una expresión constante. Sin embargo, en este caso no existe una constante con el nombre *bar*. PHP asume ahora que usted quiso decir *bar* literalmente, como la cadena *"bar"*, pero que olvidó escribir las comillas.

¿Entonces porqué está mal?

En algún momento en el futuro, el equipo de PHP puede querer usar otra constante o palabra clave, o puede que usted introduzca otra constante en su aplicación, y entonces se ve en problemas. Por ejemplo, en este momento no puede usar las palabras *empty* y *default* de esta forma, ya que son palabras clave reservadas especiales.

> **Nota:** Reiterando, al interior de un valor **string** entre comillas dobles, es válido no rodear los índices de matriz con comillas, así que *"$foo[bar]"* es válido. Consulte los ejemplos anteriores para más detalles sobre el porqué, asi como la sección sobre procesamiento de variables en cadenas.

Conversión a matriz

Para cualquiera de los tipos: **integer**, **float**, **string**, **boolean** y **resource**, si convierte un valor a un **array**, obtiene una matriz con un elemento (con índice 0), el cual es el valor escalar con el que inició.

Si convierte un **object** a una matriz, obtiene las propiedades (variables miembro) de ese objeto como los elementos de la matriz. Las claves son los nombres de las variables miembro.

Si convierte un valor **NULL** a matriz, obtiene una matriz vacía.

Comparación

Es posible comparar matrices con **array_diff()** y mediante operadores de matriz.

Ejemplos

El tipo matriz en PHP es bastante versátil, así que aquí se presentan algunos ejemplos que demuestran el poder completo de las matrices.

```
<?php
// esto
$a = array( 'color'  => 'rojo',
            'sabor'  => 'dulce',
            'forma'  => 'redonda',
            'nombre' => 'manzana',
                        4        // la clave sera 0
          );

// es completamente ñalente con
$a['color']  = 'rojo';
$a['sabor']  = 'dulce';
$a['forma']  = 'redonda';
$a['nombre'] = 'manzana';
$a[]         = 4;        // la clave sera 0

$b[] = 'a';
$b[] = 'b';
$b[] = 'c';
// resultara en la matriz array(0 => 'a' , 1 => 'b' , 2 => 'c'),
// o simplemente array('a', 'b', 'c')
?>
```

Ejemplo 11-5. Uso de array()

```
<?php
// Array como mapa de propiedades
$mapa = array( 'version'          => 4,
               'SO'               => 'Linux',
               'idioma            => 'ingles',
               'etiquetas_cortas' => true
             );

// claves estrictamente numericas
$matriz = array( 7,
                 8,
                 0,
                 156,
                 -10
               );
// esto es lo mismo que array(0 => 7, 1 => 8, ...)

$cambios = array(         10, // clave = 0
                  5    =>  6,
                  3    =>  7,
                  'a'  =>  4,
                           11, // clave = 6 (el indice entero maximo era 5)
                  '8'  =>  2, // clave = 8 (entero!)
                  '02' => 77, // clave = '02'
                  0    => 12  // el valor 10 sera reemplazado por 12
                );

// matriz vacia
$vacio = array();
?>
```

Ejemplo 11-6. Colección

```
<?php
$colores = array('rojo', 'azul', 'verde', 'amarillo');

foreach ($colores as $color) {
    echo "&iquest;Le gusta el $color?\n";
}

?>
```

Esto producirá la salida:

```
&iquest;Le gusta el rojo?
&iquest;Le gusta el azul?
&iquest;Le gusta el verde?
&iquest;Le gusta el amarillo?
```

Note que actualmente no es posible cambiar los valores de la matriz directamente en un ciclo de ese tipo. Una solución parcial es la siguiente:

Ejemplo 11-7. Colección

```
<?php
foreach ($colores as $clave => $color) {
    // no funciona:
    //$color = strtoupper($color);

    // funciona:
    $colores[$clave] = strtoupper($color);
}
print_r($colores);
?>
```

Esto genera la salida:

```
Array
(
    [0] => ROJO
    [1] => AZUL
    [2] => VERDE
    [3] => AMARILLO
)
```

Este ejemplo crea una matriz con base uno.

Ejemplo 11-8. Índice con base 1

```
<?php
$primercuarto  = array(1 => 'Enero', 'Febrero', 'Marzo');
print_r($primercuarto);
?>
```

Esto imprime:

```
Array
(
    [1] => 'Enero'
    [2] => 'Febrero'
    [3] => 'Marzo'
)
```

Ejemplo 11-9. Llenado de una matriz

```
<?php
// llenar una matriz con todos los items de un directorio
$gestor = opendir('.');
while (false !== ($archivo = readdir($gestor))) {
    $archivos[] = $archivo;
}
closedir($gestor);
?>
```

Las matrices son ordenadas. Puede también cambiar el orden usando varias funciones de ordenamiento. Vea la sección sobre funciones de matrices para más información. Puede contar el número de items en una matriz usando la función **count()**.

Ejemplo 11-10. Ordenamiento de una matriz

```php
<?php
sort($archivos);
print_r($archivos);
?>
```

Dado que el valor de una matriz puede ser cualquier cosa, también puede ser otra matriz. De esta forma es posible crear matrices recursivas y multi-dimensionales.

Ejemplo 11-11. Matrices recursivas y multi-dimensionales

```php
<?php
$frutas = array ( "frutas"  => array ( "a" => "naranja",
                                       "b" => "banano",
                                       "c" => "manzana"
                                     ),
                  "numeros" => array ( 1,
                                       2,
                                       3,
                                       4,
                                       5,
                                       6
                                     ),
                  "hoyos"   => array (      "primero",
                                       5 => "segundo",
                                            "tercero"
                                     )
                );

// Algunos ejemplos que hacen referencia a los valores de la matriz anterior
echo $frutas["hoyos"][5];    // imprime "segundo"
echo $frutas["frutas"]["a"]; // imprime "naranja"
unset($frutas["hoyos"][0]);  // elimina "primero"

// Crear una nueva matriz multi-dimensional
$jugos["manzana"]["verde"] = "bien";
?>
```

Debe advertir que la asignación de matrices siempre involucra la copia de valores. Necesita usar el operador de referencia para copiar una matriz por referencia.

```php
<?php
$matriz1 = array(2, 3);
$matriz2 = $matriz1;
$matriz2[] = 4; // $matriz2 cambia,
                // $matriz1 sigue siendo array(2, 3)

$matriz3 = &$matriz1;
$matriz3[] = 4; // ahora $matriz1 y $matriz3 son iguales
?>
```

Objetos

Inicialización de Objetos

Para inicializar un objeto, use la sentencia *new*, lo que instancia el objeto a una variable.

```php
<?php
class foo
{
    function hacer_foo()
    {
        echo "Haciendo foo.";
    }
}

$bar = new foo;
$bar->hacer_foo();
?>
```

Para una discusión completa, por favor refiérase a la sección Clases y Objetos.

Conversión a objeto

Si un objeto es convertido a un objeto, éste no es modificado. Si un valor de cualquier otro tipo es convertido a objeto, una nueva instancia de la clase *stdClass* es creada. Si el valor era nulo, la nueva instancia será vacía. Para cualquier otro valor, una variable miembro llamada *scalar* contendrá el valor.

```php
<?php
$obj = (object) 'ciao';
echo $obj->scalar;  // imprime 'ciao'
?>
```

Recurso

Un recurso es una variable especial, que contiene una referencia a un recurso externo. Los recursos son creados y usados por funciones especiales. Vea el apéndice para un listado de todas estas funciones y los tipos de recurso correspondientes.

> **Nota:** El tipo recurso fue introducido en PHP 4

Conversión a un recurso

Dado que los tipos de recurso contienen gestores especiales a archivos abiertos, conexiones con bases de datos, áreas de pintura de imágenes y cosas por el estilo, no es posible convertir cualquier valor a un recurso.

Liberación de recursos

Gracias al sistema de conteo de referencias introducido con el Motor Zend de PHP 4, se detecta automáticamente cuando un recurso ya no es referenciado (tal como en Java). Cuando este es el caso, todos los recursos que fueron usados para éste recurso se liberan por el recolector de basura. Por esta razón, rara vez se necesita liberar la memoria manualmente mediante el uso de alguna función free_result.

> **Nota:** Los enlaces persistentes con bases de datos son especiales, ellos *no* son destruidos por el recolector de basura. Vea también la sección sobre conexiones persistentes.

NULL

El valor especial **NULL** representa que una variable no tiene valor. **NULL** es el único valor posible del tipo **NULL**.

> **Nota:** El tipo null se introdujo en PHP 4

Una variable es considerada como **NULL** si

- se ha asignado la constante **NULL** a la variable.
- no ha sido definida con valor alguno.
- ha sido eliminada con **unset()**.

Sintaxis

Existe un solo valor de tipo **NULL**, y ese es la palabra clave **NULL**, insensible a mayúsculas y minúsculas.

```
<?php
$var = NULL;
?>
```

Vea también **is_null()** y **unset()**.

Pseudo-tipos usados en esta documentación

mixed

mixed indica que un parámetro puede aceptar múltiples tipos (pero no necesariamente todos).

gettype() por ejemplo aceptará todos los tipos PHP, mientras que **str_replace()** aceptará cadenas y matrices.

---

number

number indica que un parámetro puede ser **integer** o **float**.

---

callback

Algunas funciones como **call_user_func()** o **usort()** aceptan llamadas de retorno definidas por el usuario como un parámetro. Las funciones tipo llamada de retorno no sólo pueden ser funciones simples, también pueden ser métodos de objetos incluyendo métodos estáticos de clase.

Una función de PHP es simplemente pasada usando su nombre como una cadena. Puede pasar cualquier función incorporada o definida por el usuario con la excepción de **array()**, **echo()**, **empty()**, **eval()**, **exit()**, **isset()**, **list()**, **print()** y **unset()**.

Un método de un objeto instanciado es pasado como una matriz que contiene un objeto como el elemento con el índice 0 y un nombre de método como el elemento con índice 1.

Los métodos estáticos de clase pueden ser pasados también sin instanciar un objeto de esa clase al pasar el nombre de clase en lugar de un objeto como el elemento con índice 0.

Ejemplo 11-12. Ejemplos de funciones tipo llamada de retorno

```
<?php

// ejemplo simple de una llamada de retorno
function mi_llamada_de_retorno() {
    echo '&iexcl;Hola mundo!';
}
call_user_func('mi_llamada_de_retorno');

// ejemplos de un m&eacute;todo como llamada de retorno
class MiClase {
    function miMetodoDeRetorno() {
        echo '&iexcl;Hola Mundo!';
    }
}

// llamada de metodo estatico de clase sin instanciar un objeto
call_user_func(array('MiClase', 'miMetodoDeRetorno'));

// llamada a un metodo de objeto
$obj = new MiClase();
call_user_func(array(&$obj, 'miMetodoDeRetorno'));
?>
```

Manipulación de Tipos

PHP no requiere (o soporta) la definición explícita de tipos en la declaración de variables; el tipo de una variable es determinado por el contexto en el que la variable es usada. Lo que quiere decir que si asigna un valor de cadena a la variable *$var*, *$var* se convierte en una cadena. Si luego asigna un valor entero a *$var*, ésta se convierte en entera.

Un ejemplo de la conversión automática de tipos de PHP es el operador de adición '+'. Si cualquiera de los operandos es un flotante, entonces todos los operandos son evaluados como flotantes, y el resultado será un flotante. De otro modo, los operandos serán interpretados como enteros, y el resultado será también un entero. Note que este NO modifica los tipos de los operandos mismos; el único cambio está en la forma como los operandos son evaluados.

```
<?php
$foo = "0";  // $foo es una cadena (ASCII 48)
$foo += 2;   // $foo es ahora un entero (2)
$foo = $foo + 1.3;  // $foo es ahora un flotante (3.3)
$foo = 5 + "10 Cerditos"; // $foo es entero (15)
$foo = 5 + "10 Cerdos";   // $foo es entero (15)
?>
```

Si los dos últimos ejemplos lucen extraños, consulte Conversión de cadenas a números.

Si desea forzar que una variable sea evaluada como un cierto tipo, consulte la sección sobre Moldeamiento de tipos. Si desea cambiar el tipo de una variable, vea **settype()**.

Si quisiera probar cualquiera de los ejemplos en esta sección, puede usar la función **var_dump()**.

> **Nota:** El comportamiento de una conversión automática a matriz no se encuentra definido en el momento.

```
<?php
$a = "1";     // $a es una cadena
$a[0] = "f";  // Que hay de las posiciones de cadena? Que sucede?
?>
```

Ya que PHP (por razones históricas) soporta el uso de índices en cadenas mediante desplazamientos de posición usando la misma sintaxis que la indexación de matrices, el ejemplo anterior lleva a un problema: ¿debería $a convertirse en una matriz con un primer elemento "f", o debería "f" convertirse en el primer caracter de la cadena $a?

Las versiones recientes de PHP interpretan la segunda asignación como una identificación de desplazamiento de cadena, así que $a se convierte en "f", sin embargo el resultado de esta conversión automática debe considerarse indefinido. PHP 4 introdujo la nueva sintaxis de llaves para acceder a los caracteres de una cadena, use esta sintaxis en lugar de la que fue presentada anteriormente:

```
<?php
$a    = "abc"; // $a es una cadena
$a{1} = "f";   // $a es ahora "afc"
?>
```

Vea la sección llamada Acceso a cadenas por caracter para más información.

Moldeamiento de Tipos

El moldeamiento de tipos en PHP funciona de forma muy similar a como ocurre en C: el nombre del tipo deseado es escrito entre paréntesis antes de la variable que debe ser moldeada.

```
<?php
$foo = 10;   // $foo es un entero
$bar = (boolean) $foo;   // $bar es un booleano
?>
```

Los moldeamiontos permitidos son:

- (int), (integer) - moldeamiento a entero
- (bool), (boolean) - moldeamiento a booleano
- (float), (double), (real) - moldeamiento a flotante
- (string) - moldeamiento a cadena
- (array) - moldeamiento a matriz
- (object) - moldeamiento a objeto

Note que las tabulaciones y los espacios son permitidos al interior de los paréntesis, así que las siguientes expresiones son funcionalmente ñalentes:

```
<?php
$foo = (int) $bar;
$foo = ( int ) $bar;
?>
```

Nota: En lugar de moldear una variable a cadena, puede también rodear la variable de comillas dobles.

```
<?php
$foo = 10;            // $foo es un entero
$cad = "$foo";        // $cad es una cadena
$fst = (string) $foo; // $fst es tambien una cadena

// Esto imprime "son lo mismo"
if ($fst === $cad) {
    echo "son lo mismo";
}
?>
```

Puede que no sea obvio qué sucede exactamente cuando se moldea entre ciertos tipos. Para más información, consulte las secciones:

- Conversión a booleano
- Conversión a entero
- Conversión a flotante
- Conversión a cadena
- Conversión a matriz
- Conversión a objeto
- Conversión a un recurso
- Las tablas de comparación de tipos

Capítulo 12. Variables

Conceptos Básicos

En PHP las variables se representan como un signo de dólar seguido por el nombre de la variable. El nombre de la variable es sensible a minúsculas y mayúsculas.

Los nombres de variables siguen las mismas reglas que otras etiquetas en PHP. Un nombre de variable valido tiene que empezar con una letra o una raya (underscore), seguido de cualquier número de letras, números y rayas. Como expresión regular se podria expresar como: '[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*'

Nota: En nuestro caso, una letra es a-z, A-Z, y los caracteres ASCII del 127 al 255 (0x7f-0xff).

```
<?php
$var = "Bob";
$Var = "Joe";
echo "$var, $Var";      // outputs "Bob, Joe"

$4site = 'not yet';     // invalid; starts with a number
$_4site = 'not yet';    // valid; starts with an underscore
$tÃƒÂ¤yte = 'mansikka';    // valid; 'ÃƒÂ¤' is ASCII 228 (Extendido)
?>
```

En PHP3, las variables siempre se asignan por valor. Esto significa que cuando se asigna una expresión a una variable, el valor íntegro de la expresión original se copia en la variable de destino. Esto quiere decir que, por ejemplo, después e asignar el valor de una variable a otra, los cambios que se efectúen a una de esas variables no afectará a la otra. Para más información sobre este tipo de asignación, vea Expresiones.

PHP4 ofrece otra forma de asignar valores a las variables: *asignar por referencia*. Esto significa que la nueva variable simplemente referencia (en otras palabras, "se convierte en un alias de" ó "apunta a") la variable original. Los cambios a la nueva variable afectan a la original, y viceversa. Esto también significa que no se produce una copia de valores; por tanto, la asignación ocurre más rápidamente. De cualquier forma, cualquier incremento de velocidad se notará sólo en los bucles críticos cuando se asignen grandes matrices u objetos.

Para asignar por referencia, simplemente se antepone un ampersandsigno "&" al comienzo de la variable cuyo valor se está asignando (la variable fuente). Por ejemplo, el siguiente trozo de código produce la salida 'Mi nombre es Bob' dos veces:

```
<?php
$foo = 'Bob';              // Asigna el valor 'Bob' a $foo
$bar = &amp;$foo;              // Referencia $foo v&iacute;a $bar.
$bar = "Mi nombre es $bar";  // Modifica $bar...
echo $foo;                 // $foo tambi&eacute;n se modifica.
echo $bar;
?>
```

Algo importante a tener en cuenta es que sólo las variables con nombre pueden ser asignadas por referencia.

```
<?php
$foo = 25;
$bar = &amp;$foo;      // Esta es una asignaci&oacute;n v&aacute;lida.
$bar = &amp;(24 * 7);  // Inv&aacute;lida; referencia una expresi&oacute;n sin nombre.

function test() {
   return 25;
}

$bar = &amp;test();    // Inv&aacute;lida.
?>
```

Variables predefinidas

PHP proporciona una gran cantidad de variables predefinidas a cualquier script que se ejecute. De todas formas, muchas de esas variables no pueden estar completamente documentadas ya que dependen de sobre qué servidor se esté ejecutando, la versión y configuración de dicho servidor, y otros factores. Algunas de estas variables no estarán disponibles cuando se ejecute PHP desde la línea de comandos.Para obtener una lista de estas variables podeis consultar la sección Variables predefinidas reservadas.

Aviso
A partir de PHP 4.2.0, el valor por defecto de la directiva PHP register_globals es *off* (desactivada). Este es un cambio importante en PHP. Teniendo register_globals *off* afecta el conjunto de variables predefinidas disponibles en el sistema. Por ejemplo, para obtener *DOCUMENT_ROOT* se usará *$_SERVER['DOCUMENT_ROOT']* en vez de *$DOCUMENT_ROOT* ó *$_GET['id']* de la URL *http://www.example.com/test.php?id=3* en vez de *$id* ó *$_ENV['HOME']* en vez de *$HOME*. Para más información sobre este cambio, podeis consultar el apartado de configuración sobre register_globals, el capítulo sobre seguridad Usando "Register Globals" , asi como los anuncios de lanzamiento de PHP 4.1.0 y 4.2.0 El uso de las variables reservadas predefinidas en PHP, como matrices superglobales es recomendable.

A partir de PHP 4.1.0, PHP ofrece un conjunto adicional de matrices predefinidas, conteniendo variables del servidor web, el entorno y entradas del usuario. Estas nuevas matrices son un poco especiales porque son automáticamente globales. Por esta razón, son conocidas a menudo como "autoglobales" ó "superglobales". Las superglobales se mencionan más abajo; sin embargo para una lista de sus contenidos y más información sobre variables predefinidas en PHP, consultar la sección Variables predefinidas reservadas. Podreis ver como las variables predefinidas antiguas (*$HTTP_*_VARS*) todavía existen. A partir de PHP 5.0.0, las matrices de tipo "long" de variables predefinidas, se pueden desactivar con la directiva register_long_arrays.

> **Variables variables:** Las superglobales no pueden usarse como variables variables.

Si ciertas variables no son definidas en variables_order, las matrices PHP predefinidas asociadas a estas, estarán vacias.

PHP superglobales

$GLOBALS

> Contiene una referencia a cada variable disponible en el espectro de las variables del script. Las llaves de esta matriz son los nombres de las variables globales. *$GLOBALS* existe dese PHP 3.

$_SERVER

> Variables definidas por el servidor web ó directamente relacionadas con el entorno en don el script se esta ejecutando. Análoga a la antigua matriz *$HTTP_SERVER_VARS* (la cual está todavía disponible, aunque no se use).

$_GET

> Variables proporcionadas al script por medio de HTTP GET. Análoga a la antigua matriz *$HTTP_GET_VARS* (la cual está todavía disponible, aunque no se use).

$_POST

> Variables proporcionadas al script por medio de HTTP POST. Análoga a la antigua matriz *$HTTP_POST_VARS* (la cual está todavía disponible, aunque no se use).

$_COOKIE

Variables proporcionadas al script por medio de HTTP cookies. Análoga a la antigua matriz *$HTTP_COOKIE_VARS* (la cual está todavía disponible, aunque no se use).

$_FILES

Variables proporcionadas al script por medio de la subida de ficheros via HTTP . Análoga a la antigua matriz *$HTTP_POST_FILES* (la cual está todavía disponible, aunque no se use). Vea también Subiendo ficheros por método POST para más información.

$_ENV

Variables proporcionadas al script por medio del entorno. Análoga a la antigua matriz *$HTTP_ENV_VARS* (la cual está todavía disponible, aunque no se use).

$_REQUEST

Variables proporcionadas al script por medio de cuaquier mecanismo de entrada del usuario y por lo tanto no se puede confiar en ellas. La presencia y el orden en que aparecen las variables en esta matriz es definido por la directiva de configuración variables_order. Esta matriz no tiene un análogo en versiones anteriores a PHP 4.1.0. Vea también **import_request_variables()**.

Nota: Cuando se utiliza la linea de comandos, *argv* y *argc no* son incluidas aquí; estas variables se podrán encontrar en la matriz

$_SESSION

Variables registradas en la sesión del script. Análoga a la antigua matriz *$HTTP_SESSION_VARS* (la cual está todavía disponible, aunque no se use). Vea también la sección Funciones para el manejo de sessiones para más información.

Ámbito de las variables

El ámbito de una variable es el contexto dentro del que la variable está definida. La mayor parte de las variables PHP sólo tienen un ámbito simple. Este ámbito simple también abarca los ficheros incluidos y los requeridos. Por ejemplo:

```
<?php
$a = 1;
include "b.inc";
?>
```

Aquí, la variable *$a* dentro del script incluido `b.inc`. De todas formas, dentro de las funciones definidas por el usuario aparece un ámbito local a la función. Cualquier variables que se use dentro de una función está, por defecto, limitada al ámbito local de la función. Por ejemplo:

```
<?php
$a = 1; /* global scope */

function Test()
{
    echo $a; /* reference to local scope variable */
}

Test();
?>
```

Este script no producirá salida, ya que la orden echo utiliza una versión local de la variable *$a*, a la que no se ha asignado ningún valor en su ámbito. Puede que usted note que hay una pequeña diferencia con el lenguaje C, en el que las variables globales están disponibles automáticamente dentro de la función a menos que sean expresamente sobreescritas por una definición local. Esto puede causar algunos problemas, ya que la gente puede cambiar variables globales inadvertidamente. En PHP, las variables globales deben ser declaradas globales dentro de la función si van a ser utilizadas dentro de dicha función. Veamos un ejemplo:

```
<?php
$a = 1;
$b = 2;

function Sum()
{
    global $a, $b;

    $b = $a + $b;
}

Sum();
echo $b;
?>
```

El script anterior producirá la salida "3". Al declarar *$a* y *$b* globales dentro de la función, todas las referencias a tales variables se referirán a la versión global. No hay límite al número de variables globales que se pueden manipular dentro de una función.

Un segundo método para acceder a las variables desde un ámbito global es usando la matriz *$GLOBALS*. El ejemplo anterior se puede reescribir así:

```
<?php
$a = 1;
$b = 2;

function Sum()
{
    $GLOBALS["b"] = $GLOBALS["a"] + $GLOBALS["b"];
}

Sum();
echo $b;
?>
```

La matriz *$GLOBALS* es una matriz asociativa con el nombre de la variable global como clave y los contenidos de dicha variable como el valor del elemento de la matriz. *$GLOBALS* existe en cualquier ámbito, esto pasa porque $GLOBALS es una superglobal. Aqui teneis un ejemplo que demuestra el poder de las superglobales:

```
<?php
function test_global()
{
    // Most predefined variables aren't "super" and require
    // 'global' to be available to the functions local scope.
    global $HTTP_POST_VARS;

    print $HTTP_POST_VARS['name'];

    // Superglobals are available in any scope and do
    // not require 'global'.  Superglobals are available
    // as of PHP 4.1.0
    print $_POST['name'];
}
?>
```

Otra característica importante del ámbito de las variables es la variable *static*. Una variable estática existe sólo en el ámbito local de la función, pero no pierde su valor cuando la ejecución del programa abandona este ámbito. Consideremos el siguiente ejemplo:

```
<?php
function Test ()
{
    $a = 0;
    echo $a;
    $a++;
}
?>
```

Esta función tiene poca utilidad ya que cada vez que es llamada asigna a *$a* el valor *0* y representa un "0". La sentencia *$a*++, que incrementa la variable, no sirve para nada, ya que en cuanto la función termina la variable *$a* desaparece. Para hacer una función útil para contar, que no pierda la pista del valor actual del conteo, la variable *$a* debe declararse como estática:

```
<?php
function Test()
{
    static $a = 0;
    echo $a;
    $a++;
}
?>
```

Ahora, cada vez que se llame a la función Test(), se representará el valor de *$a* y se incrementará.

Las variables estáticas también proporcionan una forma de manejar funciones recursivas. Una función recursiva es la que se llama a sí misma. Se debe tener cuidado al escribir una función recursiva, ya que puede ocurrir que se llame a sí misma indefinidamente. Hay que asegurarse de implementar una forma adecuada de terminar la recursión. La siguiente función cuenta recursivamente hasta 10, usando la variable estática *$count* para saber cuándo parar:

```
<?php
function Test()
{
    static $count = 0;

    $count++;
    echo $count;
    if ($count < 10) {
        Test ();
    }
    $count--;
}
?>
```

En motor Zend 1, utilizado por *PHP4*, implementa los modificadores *static* y *global* para variables

en términos de referencias. Por ejemplo, una variable global verdadera importada dentro del ámbito de una función con *global*, crea una referencia a la variable global. Esto puede ser causa de un comportamiento inesperado, tal y como podemos comprobar en el siguiente ejemplo:

```
<?php
function test_global_ref() {
    global $obj;
    $obj = &new stdclass;
}

function test_global_noref() {
    global $obj;
    $obj = new stdclass;
}

test_global_ref();
var_dump($obj);
test_global_noref();
var_dump($obj);
?>
```

Al ejecutar este ejemplo obtendremos la siguiente salida:

```
NULL
object(stdClass)(0) {
}
```

Un comportamiento similar se aplica a *static*. Referencias no son guardadas estáticamente.

```
<?php
function &get_instance_ref() {
    static $obj;

    echo "Static object: ";
    var_dump($obj);
    if (!isset($obj)) {
        // Assign a reference to the static variable
        $obj = &new stdclass;
    }
    $obj->property++;
    return $obj;
}

function &get_instance_noref() {
    static $obj;

    echo "Static object: ";
    var_dump($obj);
    if (!isset($obj)) {
        // Assign the object to the static variable
        $obj = new stdclass;
    }
    $obj->property++;
    return $obj;
}

$obj1 = get_instance_ref();
$still_obj1 = get_instance_ref();
echo "\n";
$obj2 = get_instance_noref();
$still_obj2 = get_instance_noref();
?>
```

Al ejecutar este ejemplo obtendremos la siguiente salida:

```
Static object: NULL
Static object: NULL

Static object: NULL
Static object: object(stdClass)(1) {
  ["property"]=>
  int(1)
}
```

Este ejemplo demuestra que al asignar una referencia a una variable estática, esta no es *recordada* cuando se invoca la funcion *&get_instance_ref()* por segunda vez.

Variables variables

A veces es conveniente tener nombres de variables variables. Dicho de otro modo, son nombres de variables que se pueden establecer y usar dinámicamente. Una variable normal se establece con una sentencia como:

```
<?php
$a = "hello";
?>
```

Una variable variable toma el valor de una variable y lo trata como el nombre de una variable. En el ejemplo anterior, *hello*, se puede usar como el nombre de una variable utilizando dos signos de dólar. p.ej.

```
<?php
$$a = "world";
?>
```

En este momento se han definido y almacenado dos variables en el árbol de símbolos de PHP: *$a*, que contiene "hello", y *$hello*, que contiene "world". Es más, esta sentencia:

```
<?php
echo "$a ${$a}";
?>
```

produce el mismo resultado que:

```
<?php
echo "$a $hello";
?>
```

p.ej. ambas producen el resultado: *hello world*.

Para usar variables variables con matrices, hay que resolver un problema de ambigüedad. Si se escribe *$$a[1]* el intérprete necesita saber si nos referimos a utilizar *$a[1]* como una variable, o si se pretendía utilizar *$$a* como variable y el índice [1] como índice de dicha variable. La sintaxis para resolver esta ambigüedad es: *${$a[1]}* para el primer caso y ${$a}[1] para el segundo.

Aviso
Tener en cuenta que variables variables no pueden usarse con Matrices superglobales. Esto significa que no se pueden hacer cosas como *${$_GET}*. Si buscais un método para manejar la disponibilidad de superglobales y las antiguas *HTTP_*_VARS*, podeis intentar referiros a ellas.

Variables externas a PHP

Formularios HTML (GET y POST)

Cuando se envía un formulario a un script PHP, las variables de dicho formulario pasan a estar automáticamente disponibles en el script gracias a PHP. Por ejemplo, consideremos el siguiente formulario:

Ejemplo 12-1. Variables de formulario simples

```
<form action="foo.php" method="POST">
    Name:  <input type="text" name="username"><br>
    Email: <input type="text" name="email"><br>
    <input type="submit" name="submit" value="Submit me!">
</form>
```

Dependiendo de tu configuración y preferencias personales, existen muchas maneras de acceder a los datos de tus formularios HTML. Algunos ejemplos:

Ejemplo 12-2. Accediendo datos de un formulario simple HTML POST

```
<?php
// Available since PHP 4.1.0

   print $_POST['username'];
   print $_REQUEST['username'];

   import_request_variables('p', 'p_');
   print $p_username;

// Available since PHP 3.  As of PHP 5.0.0, these long predefined
// variables can be disabled with the register_long_arrays directive.

   print $HTTP_POST_VARS['username'];

// Available if the PHP directive register_globals = on.  As of
// PHP 4.2.0 the default value of register_globals = off.
// Using/relying on this method is not preferred.

   print $username;
?>
```

Usando un formulario GET es similar excepto en el uso de variables predefinidas, que en este caso serán del tipo GET. GET también se usa con QUERY_STRING (la información despues del símbolo '?' en una URL). Por ejemplo *http://www.example.com/test.php?id=3* contiene datos GET que son accesibles con *$_GET['id']*. Vea también $_REQUEST y **import_request_variables()**.

> **Nota:** Matrices superglobales, como *$_POST* y*$_GET*, están disponibles desde PHP 4.1.0.

Como hemos dicho, antes de PHP 4.2.0, el valor por defecto de register_globals era *on* (activado). Y, en PHP 3 estba siempre activado. La comunidad PHP anima a no confiar en esta directiva ya que es preferible asumir que tiene el valor *off* (desactivada) y programar teniendo en cuenta esto.

> **Nota:** La directiva de configuración magic_quotes_gpc afecta a valores Get, Post y Cookie, Si esta activada (on) el valor (It's "PHP!") sera convertido automaticamente a (It\'s \"PHP!\"). "Escaping" es necesario en inserciones a bases de datos. Vea también **addslashes()**, **stripslashes()** y magic_quotes_sybase.

PHP tambien entiende matrices en el contexto de variables de formularios. (vea la faq relacionada). Se puede, por ejemplo, agrupar juntas variables relacionadas ó usar esta característica para obtener valores de una entrada "select2 múltiple. Por ejemplo, vamos a mandar un formulario asi mismo y a

presentar los datos cuando se reciban:

Ejemplo 12-3. Variables de formulario más complejas

```
<?php
if ($HTTP_POST_VARS['action'] == 'submitted') {
    print '<pre>';

    print_r($HTTP_POST_VARS);
    print '<a href="'. $HTTP_SERVER_VARS['PHP_SELF'] .'">Please try again</a>';

    print '</pre>';
} else {
?>
<form action="<?php echo $HTTP_SERVER_VARS['PHP_SELF']; ?>" method="POST">
    Name:  <input type="text" name="personal[name]"><br>
    Email: <input type="text" name="personal[email]"><br>
    Beer: <br>
    <select multiple name="beer[]">
        <option value="warthog">Warthog</option>
        <option value="guinness">Guinness</option>
        <option value="stuttgarter">Stuttgarter SchwabenbrÃƒÂ¤u</option>
    </select><br>
    <input type="hidden" name="action" value="submitted">
    <input type="submit" name="submit" value="submit me!">
</form>
<?php
}
?>
```

en PHP 3, el uso de matrices de variables de formularios está limitado a matrices unidimensionales. En PHP 4, no existe esta restricción.

IMAGE SUBMIT variable names

Cuando mandamos un formulario, es posible usar una imagen en vez del botón estandar de "mandar":

```
<input type="image" src="image.gif" name="sub">
```

Cuando el usuario hace click en cualquier parte de la imagen, el formulario que la acompaña se transmitirá al servidor con dos variables adicionales, sub_x y sub_y. Estas contienen las coordenadas del click del usuario dentro de la imagen. Los más experimentados puede notar que los nombres de variable enviados por el navegador contienen un guión en vez de un subrayado (guión bajo), pero PHP convierte el guión en subrayado automáticamente.

Cookies HTTP

PHP soporta cookies de HTTP de forma transparente tal y como están definidas en en las Netscape's Spec. Las cookies son un mecanismo para almacenar datos en el navegador y así rastrear o identificar a usuarios que vuelven. Se pueden crear cookies usando la función **SetCookie()**. Las cookies son parte de la cabecera HTTP, así que se debe llamar a la función SetCookie antes de que se envíe cualquier salida al navegador. Es la misma restricción que para la función **header()**. Los datos de una cookie estan disponibles en la matriz con datos de cookies apropiada, tal como *$_COOKIE*, *$HTTP_COOKIE_VARS* y también en *$_REQUEST*. vea la función **setcookie()** para más detalles y ejemplos.

Si se quieren asignar múltiples valores a una sola cookie, basta con añadir *[]* al nombre de la cookie para definirla como una matriz. Por ejemplo:

```
<?php
  setcookie("MyCookie[foo]", "Testing 1", time()+3600);
  setcookie("MyCookie[bar]", "Testing 2", time()+3600);
?>
```

Esto creara dos cookies separadas aunque MyCookie será una matriz simple en el script. Si se quiere definir una sola cookie con valores multiples, considerar primero el uso de la función **serialize()** ó **explode()** en el valor.

Nótese que una cookie reemplazará a una cookie anterior que tuviese el mismo nombre en el navegador a menos que el camino (path) o el dominio fuesen diferentes. Así, para una aplicación de carro de la compra se podría querer mantener un contador e ir pasándolo. P.ej.

Ejemplo 12-4. Ejemplo SetCookie

```
<?php
$count++;
setcookie("count", $count, time()+3600);
setcookie("Cart[$count]", $item, time()+3600);
?>
```

Puntos en los nombres de variables de entrada

Típicamente, PHP no altera los nombres de las variables cuando se pasan a un script. De todas formas, hay que notar que el punto no es un carácter válido en el nombre de una variable PHP. Por esta razón:

```
<?php
$varname.ext;  /* nombre de variable invalido */
?>
```

Lo que el intérprete ve es el nombre de una variable $varname, seguido por el operador de concatenación, y seguido por la prueba (es decir, una cadena sin entrecomillar que no coincide con ninguna palabra clave o reservada conocida) 'ext'. Obviamente, no se pretendía que fuese este el resultado.

Por esta razón, es importante hacer notar que PHP reemplazará automáticamente cualquier punto en los nombres de variables de entrada por guiones bajos (subrayados).

Determinando los tipos de variables

Dado que PHP determina los tipos de las variables y los convierte (generalmente) según lo necesita, no siempre resulta obvio de qué tipo es una variable dada en un momento concreto. PHP incluye varias funciones que descubren de qué tipo es una variable: **gettype()**, **is_array()**, **is_float()**, **is_int()**, **is_object()**, y **is_string()**. Vea también el capítulo sobre Tipos.

Capítulo 13. Constantes

Una constante es un identificador para expresar un valor simple. Como el nombre sugiere, este valor no puede variar durante la ejecución del script. (Las constantes especiales **__FILE__** y **__LINE__** son una excepción a esto, ya que actualmente no lo soin). Una constante es sensible a mayúsculas por defecto. Por convención, los identificadores de constantes suelen declararse en mayúsculas

El nombre de una constante sigue las mismas reglas que cualquier etiqueta en PHP. Un nombre de constante válido empieza con una letra o un caracter de subrayado, seguido por cualquier número de letras, números, o subrayados. Se podrían expresar mediante la siguiente expresión regular: *[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]**

> **Nota:** Para nuestros propósitos , entenderemos como letra los carácteres a-z, A-Z, y los ASCII del 127 hasta el 255 (0x7f-0xff).

El alcanze de una constante es global, Es decir, es posible acceder a ellas sin preocuparse por el ámbito de alcance.

Sintaxis

Se puede definir una constante usando la función **define()**. Una vez definida, no puede ser modificada ni eliminada .

Solo se puede definir como constantes valores escalares (**boolean**, **integer**, **float** y **string**).

Para obtener el valor de una constante solo es necesario especificar su nombre. A diferencia de las variables, *no* se tiene que especificar el prefijo *$*. Tambien se puede utilizar la función **constant()**, para obtener el valor de una constante, en el caso de que queramos expresarla de forma dinámica Usa la función **get_defined_constants()** parar obtener una lista de todas las constantes definidas.

> **Nota:** Las contantes y las variables (globales) se encuentran en un espacio de nombres distinto. Esto implica que por ejemplo **`TRUE`** y *$TRUE* son diferentes.

Si usas una constante todavia no definida, PHP asume que estás refiriéndote al nombre de la constante en si. Se lanzará un aviso si esto sucede. Usa la función **defined()** para comprobar la existencia de dicha constante.

Estas son las diferencias entre constantes y variables:

- Las constantes no son precedidas por un símbolo de dolar (*$*)
- Las contantes solo pueden ser definidas usando la **función()** define , nunca por simple asignación
- Las constantes pueden ser definidas y accedidas sin tener en cuenta las reglas de alcanze del ámbito.
- Las constantes no pueden ser redefinidas o eliminadas despues de establecerse; y
- Las constantes solo puede albergar valores escalares

Ejemplo 13-1. Definiendo constantes

```
<?php
define("CONSTANT", "Hello world.");
echo CONSTANT; // outputs "Hello world."
echo Constant; // outputs "Constant" and issues a notice.
?>
```

Constantes predefinidas

PHP ofrece un largo número de constantes predefinidas a cualquier script en ejecución. Muchas de estas constantes, sin embargo, son creadas por diferentes extensiones, y solo estarán presentes si dichas extensiones están disponibles, bien por carga dinámica o porque has sido compiladas.

Se puede encontrar una lista de constantes predefinidas en la seccion Constantes predefinidas.

Capítulo 14. Expresiones

Las expresiones son la piedra angular de PHP. En PHP, casi cualquier cosa que escribes es una expresión. La forma más simple y ajustada de definir una expresión es "cualquier cosa que tiene un valor".

Las formas más básicas de expresiones son las constantes y las variables. Cuando escribes "$a = 5", estás asignando '5' a $a. '5', obviamente, tiene el valor 5 ó, en otras palabras '5' es una expresión con el valor 5 (en este caso, '5' es una constante entera).

Después de esta asignación, se espera que el valor de $a sea 5 también, de manera que si escribes $b = $a, se espera también que se comporte igual que si escribieses $b = 5. En otras palabras, $a es una expresión también con el valor 5. Si todo va bien, eso es exactamente lo que pasará.

Las funciones son un ejemplo algo más complejo de expresiones. Por ejemplo, considera la siguiente función:

```
<?php
function foo () {
    return 5;
}
?>
```

Suponiendo que estés familiarizado con el concepto de funciones (si no lo estás échale un vistazo al capítulo sobre funciones), asumirás que teclear *$c = foo()* es esencialmente lo mismo que escribir *$c = 5*, y has acertado. Las funciones son expresiones que valen el valor que retornan. Como foo() devuelve 5, el valor de la expresión 'foo()' es 5. Normalmente las funciones no devuelven un valor fijo, sino que suele ser calculado.

Desde luego, los valores en PHP no se limitan a enteros, y lo más normal es que no lo sean. PHP soporta tres tipos escalares: enteros, punto flotante y cadenas (los tipos escalares son aquellos cuyos valores no pueden 'dividirse' en partes menores, no como los arrays, por ejemplo). PHP también soporta dos tipos compuestos (no escalares): arrays y objetos. Se puede asignar cada uno de estos tipos de valor a variables o bien retornarse de funciones, sin ningún tipo de limitación.

Hasta aquí, los usuarios de PHP/FI 2 no deberían haber notado ningún cambio. Sin embargo, PHP lleva las expresiones mucho más allá, al igual que otros lenguajes. PHP es un lenguaje orientado a expresiones, en el sentido de que casi todo es una expresión. Considera el ejemplo anterior '$a = 5'. Es sencillo ver que hay dos valores involucrados, el valor de la constante entera '5', y el valor de $a que está siendo actualizado también a 5. Pero la verdad es que hay un valor adicional implicado aquí, y es el valor de la propia asignación. La asignación misma se evalúa al valor asignado, en este caso 5. En la práctica, quiere decir que '$a = 5', independientemente de lo que hace, es una expresión con el valor 5. De esta manera, escribir algo como '$b = ($a = 5)' es como escribir '$a = 5; $b = 5;' (un punto y coma marca el final de una instrucción). Como las asignaciones se evalúan de derecha a izquierda, puedes escribir también '$b = $a = 5'.

Otro buen ejemplo de orientación a expresiones es el pre y post incremento y decremento. Los usuarios de PHP/FI 2 y los de otros muchos lenguajes les sonará la notación variable++ y variable--. Esto son las operaciones de incremento y decremento. En PHP/FI 2, la instrucción '$a++' no tiene valor (no es una expresión), y no puedes asignarla o usarla de ningún otro modo. PHP mejora las características del incremento/decremento haciéndolos también expresiones, como en C. En PHP, como en C, hay dos tipos de incremento - pre-incremento y post-incremento. Ambos, en esencia, incrementan la variable y el efecto en la variable es idéntico. La diferencia radica en el valor de la propia expresion incremento. El preincremento , escrito '++$variable', se evalúa al valor incrementado (PHP incrementa la variable antes de leer su valor, de ahí el nombre 'preincremento'). El postincremento, escrito '$variable++', se evalúa al valor original de $variable antes de realizar el incremento (PHP incrementa la variable después de leer su valor, de ahí el nombre 'postincremento').

Un tipo muy corriente de expresiones son las expresiones de comparación. Estas expresiones se evalúan a 0 o 1, significando FALSO (**FALSE**) o VERDADERO (**TRUE**), respectivamente. PHP soporta > (mayor que), >= (mayor o igual que), == (igual que), != (distinto), < (menor que) y <= (menor o igual que). Estas expresiones se usan frecuentemente dentro de la ejecución condicional como la instrucción *if*.

El último tipo de expresiones que trataremos, es la combinación operador-asignación. Ya sabes que si quieres incrementar $a en 1, basta con escribir '$a++' o ++$a'. Pero qué pasa si quieres añadir más de 1, por ejemplo 3? Podrías escribir '$a++' múltiples veces, pero no es una forma de hacerlo ni eficiente ni cómoda. Una práctica mucho más corriente es escribir '$a = $a + 3'. '$a + 3' se evalúa al valor de $a más 3, y se asigna de nuevo a $a, lo que resulta en incrementar $a en 3. En PHP, como en otros lenguajes como C, puedes escribir esto de una forma más concisa, que con el tiempo será más clara y también fácil de entender. Añadir 3 al valor actual de $a se puede escribir como '$a += 3'. Esto quiere decir exactamente "toma el valor de $a, súmale 3, y asígnalo otra vez a $a". Además de ser más corto y claro, también resulta en una ejecución más rápida. El valor de '$a += 3', como el valor de una asignación normal y corriente, es el valor asignado. Ten en cuenta que NO es 3, sino el valor combinado de $a más 3 (ése es el valor asignado a $a). Cualquier operación binaria puede ser usada en forma de operador-asignación, por ejemplo '$a -= 5' (restar 5 del valor de $a), '$b *= 7' (multiplicar el valor de $b por 7), etc.

Hay otra expresión que puede parecer extraña si no la has visto en otros lenguaes, el operador condicional ternario:

```
<?php
$first ? $second : $third
?>
```

Si el valor de la primera subexpresión es verdadero (distinto de cero), entonces se evalúa la segunda subexpresión, si no, se evalúa la tercera y ése es el valor.

El siguiente ejemplo te ayudará a comprender un poco mejor el pre y post incremento y las expresiones en general:

```
<?php
function double($i) {
    return $i*2;
}
$b = $a = 5;        /* asignar el valor cinco a las variables $a y $b */
$c = $a++;          /* postincremento, asignar el valor original de $a (5) a $c */
$e = $d = ++$b;     /* preincremento, asignar el valor incrementado de $b (6) a
                       $d y a $e */

/* en este punto, tanto $d como $e son iguales a 6 */

$f = double($d++);  /* asignar el doble del valor de $d antes
                       del incremento, 2*6 = 12 a $f */
$g = double(++$e);  /* asignar el doble del valor de $e despu&eacute;s
                       del incremento, 2*7 = 14 a $g */
$h = $g += 10;      /* primero, $g es incrementado en 10 y termina valiendo 24.
                       despu&eacute;s el valor de la asignaci&oacute;n (24) se asigna a
                       y $h tambi&eacute;n acaba valiendo 24. */
?>
```

Al principio del capítulo hemos dicho que describiríamos los distintos tipos de instrucciones y, como prometimos, las expresiones pueden ser instrucciones. Sin embargo, no todas las expresiones son instrucciones. En este caso, una instrucción tiene la forma 'expr' ';', es decir, una expresión seguida de un punto y coma. En '$b=$a=5;', $a=5 es una expresión válida, pero no es una instrucción en sí misma. Por otro lado '$b=$a=5:' sí es una instrucción válida.

Una última cosa que vale la pena mencionar, es el valor booleano de las expresiones. En muchas ocasiones, principalmente en condicionales y bucles, no estás interesado en el valor exacto de la expresión, sino únicamente si es **TRUE** ó **FALSE** Las constantes **TRUE** y **FALSE** son los dos posibles valores booleanos. Cuando es necesario, una expresión se puede transformar automáticamente al tipo booleano. Consultar la sección sobre type-casting para obtener detalles de como se hace.

PHP brinda una completa y potente implementación de expresiones, y documentarla enteramente está más allá del objetivo de este manual. Los ejemplos anteriores, deberían darte una buena idea de qué son las expresiones y cómo construir expresiones útiles. A lo largo del resto del manual, escribiremos *expr* para indicar una expresión PHP válida.

Capítulo 15. Operadores

Un operador es algo a lo que usted entrega uno o más valores (o expresiones, en jerga de programación) y produce otro valor (de modo que la construcción misma se convierte en una expresión). Así que puede pensar sobre las funciones o construcciones que devuelven un valor (como print) como operadores, y en aquellas que no devuelven nada (como echo) como cualquier otra cosa.

Existen tres tipos de operadores. En primer lugar se encuentra el operador unario, el cual opera sobre un único valor, por ejemplo ! (el operador de negación) o ++ (el operador de incremento). El segundo grupo se conoce como operadores binarios; éste grupo contiene la mayoría de operadores que soporta PHP, y una lista se encuentra disponible más adelante en la sección Precedencia de Operadores.

El tercer grupo consiste del operador ternario: ?:. Éste debe ser usado para seleccionar entre dos expresiones, en base a una tercera, en lugar de seleccionar dos sentencias o rutas de ejecución. Rodear las expresiones ternarias con paréntesis es una muy buena idea.

Precedencia de Operadores

La precedencia de un operador indica qué tan "cerca" se agrupan dos expresiones. Por ejemplo, en la expresión *1 + 5 * 3*, la respuesta es *16* y no *18*, ya que el operador de multiplicación ("*") tiene una mayor precedencia que el operador de adición ("+"). Los paréntesis pueden ser usados para marcar la precedencia, si resulta necesario. Por ejemplo: *(1 + 5) * 3* evalúa a *18*. Si la precedencia de los operadores es la misma, se utiliza una asociación de izquierda a derecha.

La siguiente tabla lista la precedencia de los operadores, con aquellos de mayor precedencia listados al comienzo de la tabla. Los operadores en la misma línea tienen la misma precedencia, en cuyo caso su asociatividad decide el orden para evaluarlos.

Tabla 15-1. Precedencia de Operadores

Asociatividad	Operadores
no-asociativo	new
derecha	[
no-asociativos	++ --
no-asociativos	! ~ - (int) (float) (string) (array) (object) @
izquierda	* / %
izquierda	+ - .
izquierda	<< >>
no-asociativos	< <= > >=
no-asociativos	== != === !==
izquierda	&
izquierda	^
izquierda	|
izquierda	&&
izquierda	||
izquierda	? :
derecha	= += -= *= /= .= %= &= |= ^= <<= >>=
izquierda	and
izquierda	xor
izquierda	or
izquierda	,

La asociatividad de izquierda quiere decir que la expresión es evaluada desde la izquierda a la derecha, la asociatividad de derecha quiere decir lo contrario.

Ejemplo 15-1. Asociatividad

```php
<?php
$a = 3 * 3 % 5; // (3 * 3) % 5 = 4
$a = true ? 0 : true ? 1 : 2; // (true ? 0 : true) ? 1 : 2 = 2

$a = 1;
$b = 2;
$a = $b += 3; // $a = ($b += 3) -> $a = 5, $b = 5
?>
```

Use paréntesis para incrementar la legibilidad del código.

> **Nota:** Aunque *!* tiene una mayor precedencia que =, PHP permitirá aun expresionas similares a la siguiente: *if (!$a = foo())*, en cuyo caso la salida de *foo()* va a dar a *$a*.

Operadores de Aritmética

¿Recuerda la aritmética básica del colegio? Éstos operadores funcionan tal como aquéllos.

Tabla 15-2. Operadores de Aritmética

Ejemplo	Nombre	Resultado
-$a	Negación	El opuesto de $a.
$a + $b	Adición	Suma de $a y $b.
$a - $b	Substracción	Diferencia entre $a y $b.
$a * $b	Multiplicación	Producto de $a y $b.
$a / $b	División	Cociente de $a y $b.
$a % $b	Módulo	Resto de $a dividido por $b.

El operador de división ("/") devuelve un valor flotante en todos los casos, incluso si los dos operandos son enteros (o cadenas que son convertidas a enteros).

> **Nota:** El resto de *$a % $b* es negativo para valores negativos de *$a*.

Vea también la página del manual sobre Funciones matemáticas.

Operadores de Asignación

El operador básico de asignación es "=". A primera vista, usted podría pensar en él como "es igual a". No lo haga. Lo que quiere decir en realidad es que el operando de la izquierda recibe el valor de la expresión a la derecha (es decir, "se define a").

El valor de una expresión de asignación es el valor que se asigna. Es decir, el valor de "$a = 3" es 3. Esto le permite hacer una que otra cosa curiosa:

```php
<?php

$a = ($b = 4) + 5; // $a es igual a 9 ahora, y $b ha sido definido a 4.

?>
```

En conjunto con el operador básico de asignación, existen "operadores combinados" para todas las operaciones de aritmética binaria y de cadenas, que le permiten usar un valor en una expresión y luego definir su valor como el resultado de esa expresión. Por ejemplo:

```php
<?php

$a = 3;
$a += 5; // define $a como 8, como si hubiesemos dicho: $a = $a + 5;
$b = "&iexcl;Hola ";
$b .= "a todos!"; // define $b como "&iexcl;Hola a todos!", tal como $b = $b . "a todos

?>
```

Note que la asignación copia la variable original en la nueva (asignación por valor), de modo que cualquier cambio a una no afecta a la otra. Esto puede resultar de importancia si necesita copiar algo como una matriz de gran tamaño al interior de un ciclo reducido. A partir de PHP4, es soportada la asignación por referencia, usando la sintaxis `$var = &$otra_var;`, pero esto no es posible en PHP3. 'Asignación por referencia' quiere decir que ambas variables terminan apuntando a los mismos datos y que nada es realmente copiado. Para aprender más sobre las referencias, por favor refiérase a las Referencias explicadas.

Operadores Bit a Bit

Los operadores bit a bit le permiten activar o desactivar bits individuales de un entero. Si los parámetros tanto a la izquierda y a la derecha son cadenas, el operador bit a bit trabajará sobre los valores ASCII de los caracteres.

```php
<?php
echo 12 ^ 9; // Imprime '5'

echo "12" ^ "9"; // Imprime el caracter Backspace (ascii 8)
                 // ('1' (ascii 49)) ^ ('9' (ascii 57)) = #8

echo "hallo" ^ "hello"; // Imprime los valores ascii #0 #4 #0 #0 #0
                        // 'a' ^ 'e' = #4
?>
```

Tabla 15-3. Operadores Bit a Bit

Ejemplo	Nombre	Resultado
$a & $b	Y	Los bits que están activos tanto en $a como en $b son activados.
$a | $b	O	Los bits que están activos ya sea en $a o en $b son activados.
$a ^ $b	O exclusivo (Xor)	Los bitos que estén activos en $a o $b, pero no en ambos, son activados.
~ $a	No	Los bits que estén activos en $a son desactivados, y vice-versa.
$a << $b	Desplazamiento a izquierda	Desplaza los bits de $a, $b pasos a la izquierda (cada paso quiere decir "multiplicar por dos")
$a >> $b	Desplazamiento a derecha	Desplaza los bits de $a, $b pasos a la derecha (cada paso quiere decir "dividir por dos")

Aviso
No realice desplazamientos a derecha para más de 32 bits en sistemas de 32 bits. No realice desplazamientos a izquierda en caso de que resulte en un número de más de 32 bits.

Operadores de Comparación

Los operadores de comparación, como su nombre indica, le permiten comparar dos valores. Puede que también se encuentre interesado en consultar las tablas de comparación de tipos, ya que éstas muestran ejemplos de varios tipos de comparaciones relacionadas con tipos.

Tabla 15-4. Operadores de Comparación

Ejemplo	Nombre	Resultado
$a == $b	Igual	**TRUE** si $a es igual a $b.
$a === $b	Idéntico	**TRUE** si $a es igual a $b, y son del mismo tipo. (A partir de PHP 4)
$a != $b	Diferente	**TRUE** si $a no es igual a $b.
$a <> $b	Diferente	**TRUE** si $a no es igual a $b.
$a !== $b	No idénticos	**TRUE** si $a no es igual a $b, o si no son del mismo tipo. (A partir de PHP 4)
$a < $b	Menor que	**TRUE** si $a es escrictamente menor que $b.
$a > $b	Mayor que	**TRUE** si $a es estrictamente mayor que $b.
$a <= $b	Menor o igual que	**TRUE** si $a es menor o igual que $b.
$a >= $b	Mayor o igual que	**TRUE** si $a es mayor o igual que $b.

Si compara un entero con una cadena, la cadena es convertida a un número. Si compara dos cadenas numéricas, ellas son comparadas como enteros. Estas reglas también se aplican a la sentencia switch.

```
<?php
var_dump(0 == "a"); // 0 == 0 -> true
var_dump("1" == "01"); // 1 == 1 -> true

switch ("a") {
case 0:
    echo "0";
    break;
case "a": // nunca se ejecuta ya que "a" ya ha coincidido con 0
    echo "a";
    break;
}
?>
```

Otro operador condicional es el operador "?:" (o ternario).

```php
<?php
// Ejemplo de uso de: el Operador Ternario
$accion = (empty($_POST['accion'])) ? 'predeterminada' : $_POST['accion'];

// La sentencia anterior es identica a este bloque if/else
if (empty($_POST['accion'])) {
    $accion = 'predeterminada';
} else {
    $accion = $_POST['accion'];
}
?>
```

La expresión *(expr1) ? (expr2) : (expr3)* evalúa a *expr2* si *expr1* evalúa a **TRUE**, y *expr3* si *expr1* evalúa a **FALSE**.

Vea también **strcasecmp()**, **strcmp()**, Operadores de matriz, y la sección del manual sobre Tipos.

Operadores de Control de Errores

PHP ofrece soporte para un operador de control de errores: el signo de arroba (@). Cuando es colocado al comienzo de una expresión en PHP, cualquier mensaje de error que pudiera generarse a causa de esa expresión será ignorado.

Si la característica *track_errors* está habilitada, cualquier mensaje de error generado por la expresión será almacenado en la variable $php_errormsg. La variable será sobrescrita en cada instancia de error, así que realice sus chequeos de forma temprana si quiere usarla.

```php
<?php
/* Error intencional de archivo */
$mi_archivo = @file ('archivo_que_no_existe') or
    die ("La apertura de archivo ha fallado: el error fue '$php_errormsg'");

// esto funciona con cualquier expresion, no solo con funciones:
$valor = @$cache[$llave];
// no producira una anotacion si el indice $llave no existe.

?>
```

> **Nota:** El operador @ trabaja sólo sobre expresiones. Una simple regla de oro es: si usted puede tomar el valor de algo, entonces puede usar el operador @ sobre ese algo. Por ejemplo, puede usarlo al inicio de variables, llamadas a funciones y sencencias **include()**, constantes, y así sucesivamente. No puede usarlo sobre definiciones de función o clase, ni sobre estructuras condicionales como *if* y *foreach*, y así sucesivamente.

Vea también **error_reporting()** y la sección del manual sobre funciones de Gestión de Errores y Registros.

> **Nota:** El operador de prefijo "@" para control de errores no deshabilitará los mensajes que son resultado de errores en la fase de análisis sintáctico.

Aviso
En la actualidad, el operador de prefijo "@" para control de errores deshabilitará incluso el reporte de errores en casos de fallos críticos que terminarán la ejecución del script. Entre otras cosas, esto quiere decir que si usa "@" para eliminar los errores de una cierta función, y ésta no se encuentra disponible o ha sido escrita de forma incorrecta, el script se detendrá en ese punto sin dar indicación alguna del motivo.

Operadores de ejecución

PHP soporta un operador de ejecución: las comillas invertidas (``). ¡Note que no se trata de comillas sencillas! PHP intentará ejecutar el contenido entre las comillas como si se tratara de un comando del intérprete de comandos; su salida será devuelta (es decir, no será simplemente volcada como salida; puede ser asignada a una variable). El uso del operador de comillas invertidas es idéntico al de **shell_exec()**.

```
<?php
$salida = `ls -al`;
echo "<pre>$salida</pre>";
?>
```

> **Nota:** El operador de comillas invertidas es deshabilitado cuando se encuentra activo safe mode o cuando se deshabilita **shell_exec()**.

Vea también la sección del manual sobre funciones de Ejecución de Programas, **popen()** **proc_open()**, y Uso de PHP desde la línea de comandos.

Operadores de Incremento/Decremento

PHP ofrece soporte de operadores de pre- y post-incremento y decremento, estilo-C.

Tabla 15-5. Operadores de Incremento/decremento

Ejemplo	Nombre	Efecto
++$a	Pre-incremento	Incrementa $a en uno, y luego devuelve $a.
$a++	Post-incremento	Devuelve $a, y luego incrementa $a en uno.
--$a	Pre-decremento	Decrementa $a en uno, luego devuelve $a.
$a--	Post-decremento	Devuelve $a, luego decrementa $a en uno.

Aquí hay un script sencillo de ejemplo:

```
<?php
echo "<h3>Postincremento</h3>";
$a = 5;
echo "Debe ser 5: " . $a++ . "<br />\n";
echo "Debe ser 6: " . $a . "<br />\n";

echo "<h3>Preincremento</h3>";
$a = 5;
echo "Debe ser 6: " . ++$a . "<br />\n";
echo "Debe ser 6: " . $a . "<br />\n";

echo "<h3>Postdecremento</h3>";
$a = 5;
echo "Debe ser 5: " . $a-- . "<br />\n";
echo "Debe ser 4: " . $a . "<br />\n";

echo "<h3>Predecremento</h3>";
$a = 5;
echo "Debe ser 4: " . --$a . "<br />\n";
echo "Debe ser 4: " . $a . "<br />\n";
?>
```

PHP sigue la convención de Perl cuando trabaja con operaciones aritméticas sobre variables de caracter, y no la convención de C. Por ejemplo, en Perl 'Z'+1 se convierte en 'AA', mientras que en C 'Z'+1 se convierte en '[' (ord('Z') == 90, ord('[') == 91). Note que las variables de caracter pueden ser incrementadas pero no decrementadas.

Ejemplo 15-2. Operaciones Aritméticas sobre Variables de Caracter

```
<?php
$i = 'W';
for($n=0; $n<6; $n++)
  echo ++$i . "\n";

/*
  Produce una salida similar a la siguiente:

X
Y
Z
AA
AB
AC

*/
?>
```

Incrementar o decrementar valores booleanos no tiene efecto.

Operadores de Lógica

Tabla 15-6. Operadores de Lógica

Ejemplo	Nombre	Resultado
$a and $b	Y	**TRUE** si tanto $a como $b son **TRUE**.
$a or $b	O	**TRUE** si cualquiera de $a o $b es **TRUE**.
$a xor $b	O exclusivo (Xor)	**TRUE** si $a o $b es **TRUE**, pero no ambos.
! $a	No	**TRUE** si $a no es **TRUE**.
$a && $b	Y	**TRUE** si tanto $a como $b son **TRUE**.
$a || $b	O	**TRUE** si cualquiera de $a o $b es **TRUE**.

La razón para tener las dos variaciones diferentes de los operadores "and" y "or" es que ellos operan con precedencias diferentes. (Vea Precedencia de Operadores.)

Operadores de Cadena

Existen dos operadores para datos tipo **string**. El primero es el operador de concatenación ('.'), el cual devuelve el resultado de concatenar sus argumentas a lado derecho e izquierdo. El segundo es el operador de asignación sobre concatenación ('.='), el cual adiciona el argumento del lado derecho al argumento en el lado izquierdo. Por favor consulte Operadores de Asignación para más información.

```
<?php
$a = "&iexcl;Hola ";
$b = $a . "Mundo!"; // ahora $b contiene "&iexcl;Hola Mundo!"

$a = "&iexcl;Hola ";
$a .= "Mundo!";     // ahora $a contiene "&iexcl;Hola Mundo!"
?>
```

Vea también las secciones del manual sobre el tipo String y las funciones de Cadenas.

Operadores de Matrices

Tabla 15-7. Operadores de Matrices

Ejemplo	Nombre	Resultado
$a + $b	Unión	Unión de $a y $b.
$a == $b	Igualdad	**TRUE** si $a y $b tienen los mismos elementos.
$a === $b	Identidad	**TRUE** si $a y $b tienen los mismos elementos en el mismo orden.
$a != $b	No-igualdad	**TRUE** si $a no es igual a $b.
$a <> $b	No-igualdad	**TRUE** si $a no es igual a $b.
$a !== $b	No-identidad	**TRUE** si $a no es idéntico a $b.

El operador + adiciona la matriz del lado derecho a aquél al lado izquierdo, al mismo tiempo que cualquier llave duplicada NO es sobrescrita.

```
<?php
$a = array("a" => "manzana", "b" => "banano");
$b = array("a" => "pera", "b" => "fresa", "c" => "cereza");

$c = $a + $b; // Union de $a y $b
echo "Uni&oacute;n de \$a y \$b: \n";
var_dump($c);

$c = $b + $a; // Union de $b y $a
echo "Uni&oacute;n de \$b y \$a: \n";
var_dump($c);
?>
```

Cuando sea ejecutado, este script producirá la siguiente salida:

```
Uni&oacute;n de $a y $b:
array(3) {
  ["a"]=>
  string(7) "manzana"
  ["b"]=>
  string(6) "banano"
  ["c"]=>
  string(6) "cereza"
}
Uni&oacute;n de $b y $a:
array(3) {
  ["a"]=>
  string(4) "pera"
  ["b"]=>
  string(5) "fresa"
  ["c"]=>
  string(6) "cereza"
}
```

Los elementos de las matrices son considerados ñalentes en la comparación si éstos tienen la misma clave y valor.

Ejemplo 15-3. Comparación de matrices

```
<?php
$a = array("manzana", "banano");
$b = array(1 => "banano", "0" => "manzana");

var_dump($a == $b); // bool(true)
var_dump($a === $b); // bool(false)
?>
```

Vea también las secciones del manual sobre el tipo Array y funciones de Matrices.

Operadores de Tipo

PHP tiene un operador único de tipo: *instanceof*. *instanceof* es usado para determinar si un objeto dado es de una clase de objeto especificada.

El operador *instanceof* fue introducido en PHP 5. Antes de esta versión, **is_a()** era utilizado, pero **is_a()** ha sido marcado como obsoleto desde entonces en favor de *instanceof*.

```
<?php
class A { }
class B { }

$cosa = new A;

if ($cosa instanceof A) {
    echo 'A';
}
if ($cosa instanceof B) {
    echo 'B';
}
?>
```

Dado que *$cosa* es un **object** de tipo A, pero no de tipo B, sólo el bloque dependiente del tipo A será ejecutado:

```
A
```

Vea también **get_class()** e **is_a()**.

Capítulo 16. Estructuras de Control

Todo script PHP se compone de una serie de sentencias. Una sentencia puede ser una asignación, una llamada a función, un bucle, una sentencia condicional e incluso una sentencia que no haga nada (una sentencia vacía). Las sentencias normalmente acaban con punto y coma. Además, las sentencias se pueden agrupar en grupos de sentencias encapsulando un grupo de sentencias con llaves. Un grupo de sentencias es también una sentencia. En este capítulo se describen los diferentes tipos de sentencias.

---

if

La construcción *if* es una de las más importantes características de muchos lenguajes, incluido PHP. Permite la ejecución condicional de fragmentos de código. PHP caracteriza una estructura *if* que es similar a la de C:

```
<?php
if (expr)
    sentencia
?>
```

Como se describe en la sección sobre expresiones, *expr* se evalúa a su valor condicional (boolean). Si *expr* se evalúa como **TRUE**, PHP ejecutará la *sentencia*, y si se evalúa como **FALSE** - la ignorará. Se puede encontrar más información sobre los valores evaluados como **FALSE** en la sección Convirtiendo a un valor condicional (boolean)'.

El siguiente ejemplo mostraría `a es mayor que b` si *$a* fuera mayor que *$b*:

```
<?php
if ($a > $b)
   print "a es mayor que b";
?>
```

A menudo, se desea tener más de una sentencia ejecutada de forma condicional. Por supuesto, no hay necesidad de encerrar cada sentencia con una cláusula *if*. En vez de eso, se pueden agrupar varias sentencias en un grupo de sentencias. Por ejemplo, este código mostraría `a es mayor que b` si *$a* fuera mayor que *$b*, y entonces asignaría el valor de *$a* a *$b*:

```
<?php
 if ($a > $b) {
     print "a es mayor que b";
     $b = $a;
 }
?>
```

Las sentencias *if* se pueden anidar indefinidamente dentro de otras sentencias *if*, lo cual proporciona una flexibilidad completa para ejecuciones condicionales en las diferentes partes de tu programa.

---

else

A menudo queremos ejecutar una sentencia si se cumple una cierta condicion, y una sentencia distinta si la condición no se cumple. Esto es para lo que sirve *else*. *else* extiende una sentencia *if* para ejecutar una sentencia en caso de que la expresión en la sentencia *if* se evalúe como **FALSE**. Por ejemplo, el siguiente código mostraría `a es mayor que b` si *$a* fuera mayor que *$b*, y `a`

`NO es mayor que b` en cualquier otro caso:

```
<?php
 if ($a > $b) {
     print "a es mayor que b";
 } else {
     print "a NO es mayor que b";
 }
?>
```

La sentencia *else* se ejecuta solamente si la expresión *if* se evalúa como **`FALSE`**, y si hubiera alguna expresión *elseif* - sólo si se evaluaron también a **`FALSE`** (Ver elseif).

---

elseif

elseif, como su nombre sugiere, es una combinación de *if* y *else*. Como *else*, extiende una sentencia *if* para ejecutar una sentencia diferente en caso de que la expresión *if* original se evalúa como **`FALSE`**. No obstante, a diferencia de *else*, ejecutará esa expresión alternativa solamente si la expresión condicional *elseif* se evalúa como **`TRUE`**. Por ejemplo, el siguiente código mostraría `a es mayor que b`, `a es igual a b` o `a es menor que b`:

```
<?php
 if ($a > $b) {
     print "a es mayor que b";
 } elseif ($a == $b) {
     print "a es igual que b";
 } else {
     print "a es mayor que b";
 }
?>
```

Puede haber varios *elseif*s dentro de la misma sentencia *if*. La primera expresión *elseif* (si hay alguna) que se evalúe como **`TRUE`** se ejecutaría. En PHP, también se puede escribir 'else if' (con dos palabras) y el comportamiento sería idéntico al de un 'elseif' (una sola palabra). El significado sintáctico es ligeramente distinto (si estas familiarizado con C, es el mismo comportamiento) pero la línea básica es que ambos resultarían tener exactamente el mismo comportamiento.

La sentencia *elseif* se ejecuta sólo si la expresión *if* precedente y cualquier expresión *elseif* precedente se evalúan como **`FALSE`**, y la expresión *elseif* actual se evalúa como **`TRUE`**.

Sintaxis Alternativa de Estructuras de Control

PHP ofrece una sintaxis altenativa para alguna de sus estructuras de control; a saber, *if*, *while*, *for*, y *switch*. En cada caso, la forma básica de la sintaxis alternativa es cambiar abrir-llave por dos puntos (:) y cerrar-llave por *endif;*, *endwhile;*, *endfor;*, or *endswitch;*, respectivamente.

```
<?php if ($a==5): ?>
 A es igual a 5
 <?php endif; ?>
```

En el ejemplo de arriba, el bloque HTML "A es igual 5" se anida dentro de una sentencia *if* escrita en la sintaxis alternativa. El bloque HTML se mostraría solamente si $a fuera igual a 5.

La sintaxis alternativa se aplica a *else* y también a *elseif*. La siguiente es una estructura *if* con *elseif* y *else* en el formato alternativo:

```
<?php
 if ($a == 5):
     print "a es igual a 5";
     print "...";
 elseif ($a == 6):
     print "a es igual a 6";
     print "!!!";
 else:
     print "a no es ni 5 ni 6";
 endif;
?>
```

Mirar también while, for, e if para más ejemplos.

---

while

Los bucles *while* son los tipos de bucle más simples en PHP. Se comportan como su contrapartida en C. La forma básica de una sentencia *while* es:

```
while (expr) sentencia
```

El significado de una sentencia *while* es simple. Le dice a PHP que ejecute la(s) sentencia(s) anidada(s) repetidamente, mientras la expresión *while* se evalúe como **TRUE**. El valor de la expresión es comprobado cada vez al principio del bucle, así que incluso si este valor cambia durante la ejecución de la(s) sentencia(s) anidada(s), la ejecución no parará hasta el fin de la iteración (cada vez que PHP ejecuta las sentencias en el bucle es una iteración). A veces, si la expresión *while* se evalúa como **FALSE** desde el principio de todo, la(s) sentencia(s) anidada(s) no se ejecutarán ni siquiera una vez.

Como con la sentencia *if*, se pueden agrupar multiples sentencias dentro del mismo bucle *while* encerrando un grupo de sentencias con llaves, o usando la sintaxis alternativa:

```
while (expr): sentencia ... endwhile;
```

Los siguientes ejemplos son idénticos, y ambos imprimen números del 1 al 10:

```
<?php
/* ejemplo 1 */

 $i = 1;
 while ($i <= 10) {
     print $i++;  /* el valor impreso ser&iacute;a
                     $i antes del incremento
                     (post-incremento) */
 }

 /* ejemplo 2 */

 $i = 1;
 while ($i <= 10):
     print $i;
     $i++;
 endwhile;
?>
```

---

do..while

Los bucles *do..while* son muy similares a los bucles *while*, excepto que las condiciones se comprueban al final de cada iteración en vez de al principio. La principal diferencia frente a los

bucles regulares *while* es que se garantiza la ejecución de la primera iteración de un bucle *do..while* (la condición se comprueba sólo al final de la iteración), mientras que puede no ser necesariamente ejecutada con un bucle *while* regular (la condición se comprueba al principio de cada iteración, si esta se evalúa como **FALSE** desde el principio la ejecución del bucle finalizará inmediatamente).

Hay una sola sintaxis para los bucles *do..while*:

```
<?php
 $i = 0;
 do {
     print $i;
 } while ($i>0);
?>
```

El bucle de arriba se ejecutaría exactamente una sola vez, después de la primera iteración, cuando la condición se comprueba, se evalúa como **FALSE** ($i no es más grande que 0) y la ejecución del bucle finaliza.

Los usuarios avanzados de C pueden estar familiarizados con un uso distinto del bucle *do..while*, para permitir parar la ejecución en medio de los bloques de código, encapsulandolos con *do..while* (0), y usando la sentencia *break*. El siguiente fragmento de código demuestra esto:

```
<?php
do {
     if ($i < 5) {
         print "i no es lo suficientemente grande";
         break;
     }
     $i *= $factor;
     if ($i < $minimum_limit) {
         break;
     }
     print "i es correcto";
     /* procesa i */
 } while(0);
?>
```

No se preocupes si no entiende esto completamente o en absoluto. Se pueden codificar archivos de comandos e incluso archivos de comandos potentes sin usar esta 'propiedad'.

---

for

Los bucles *for* son los bucles más complejos en PHP. Se comportan como su contrapartida en C. La sintaxis de un bucle *for* es:

```
for (expr1; expr2; expr3) sentencia
```

La primera expresión (*expr1*) se evalúa (ejecuta) incondicionalmente una vez al principio del bucle.

Al comienzo de cada iteración, se evalúa *expr2* . Si se evalúa como **TRUE**, el bucle continúa y las sentencias anidadas se ejecutan. Si se evalúa como **FALSE**, la ejecución del bucle finaliza.

Al final de cada iteración, se evalúa (ejecuta) *expr3*.

Cada una de las expresiones puede estar vacía. Que *expr2* esté vacía significa que el bucle debería correr indefinidamente (PHP implicitamente lo considera como **TRUE**, al igual que C). Esto puede que no sea tan inútil como se podría pensar, puesto que a menudo se quiere salir de un bucle usando una sentencia *break* condicional en vez de usar la condición de *for*.

Considera los siguientes ejemplos. Todos ellos muestran números del 1 al 10:

```
<?php
 /* ejemplo 1 */

 for ($i = 1; $i <= 10; $i++) {
     print $i;
 }

 /* ejemplo 2 */

 for ($i = 1; ;$i++) {
     if ($i > 10) {
         break;
     }
     print $i;
 }

 /* ejemplo 3 */

 $i = 1;
 for (;;) {
     if ($i > 10) {
         break;
     }
     print $i;
     $i++;
 }

 /* ejemplo 4 */

 for ($i = 1; $i <= 10; print $i, $i++) ;
?>
```

Por supuesto, el primer ejemplo parece ser el mas elegante (o quizás el cuarto), pero uno puede descubrir que ser capaz de usar expresiones vacías en bucles *for* resulta útil en muchas ocasiones.

PHP también soporta la "sintaxis de dos puntos" alternativa para bucles *for*.

```
for (expr1; expr2; expr3): sentencia; ...; endfor;
```

Otros lenguajes poseen una sentencia *foreach* para traducir un array o una tabla hash. PHP3 no posee tal construcción; PHP4 sí (ver foreach). En PHP3, se puede combinar while con las funciones **list()** y **each()** para conseguir el mismo efecto. Mirar la documentación de estas funciones para ver un ejemplo.

---

foreach

PHP 4 (PHP3 no) incluye una construcción *foreach*, tal como perl y algunos otros lenguajes. Esto simplemente da un modo fácil de iterar sobre matrices. *foreach* funciona solamente con matrices y devolverá un error si se intenta utilizar con otro tipo de datos ó variables no inicializadas. Hay dos sintaxis; la segunda es una extensión menor, pero útil de la primera:

```
foreach(expresion_array as $value) sentencia
foreach(expresion_array as $key => $value) sentencia
```

La primera forma recorre el array dado por *expresion_array*. En cada iteración, el valor del elemento actual se asigna a *$value* y el puntero interno del array se avanza en una unidad (así en el siguiente paso, se estará mirando el elemento siguiente).

La segunda manera hace lo mismo, salvo que la clave del elemento actual será asignada a la variable

$key en cada iteración.

> **Nota:** Cuando *foreach* comienza su primera ejecución, el puntero interno a la matriz se reinicia automáticamente al primer elemento de la matriz. Esto significa que no se necesita llamar a **reset()** antes de un bucle *foreach*.
>
> **Nota:** Hay que tener en cuenta que*foreach* trabaja con una copia de la matriz especificada y no la lista en si, por ello el puntero de la lista no es modificado como en la función **each()**, y los cambios en el elemento de la matriz retornado no afectan a la matriz original. De todas maneras el puntero interno a la matriz original *avanza* al procesar la matriz. suponiendo que bucle foreach se ejecuta hasta el final, el puntero interno a la matriz estar/aacute; al final de la matriz.
>
> **Nota:** *foreach* no soporta la caracteristica de suprimir mensajes de error con '@'.

Puede haber observado que las siguientes son funcionalidades idénticas:

```
<?php
$arr = array("one", "two", "three");
reset ($arr);
while (list(, $value) = each ($arr)) {
    echo "Value: $value<br>\n";
}

foreach ($arr as $value) {
    echo "Value: $value<br>\n";
}
?>
```

Las siguientes también son funcionalidades idénticas:

```
<?php
reset( $arr );
while( list( $key, $value ) = each( $arr ) ) {
   echo "Key: $key; Valor: $value&lt;br&gt;\n";
}

foreach( $arr as $key => $value ) {
   echo "Key: $key; Valor: $value&lt;br&gt;\n";
}
?>
```

Algunos ejemplos más para demostrar su uso:

```
<?php
/* foreach ejemplo 1: s&oacute;lo valor*/
$a = array(1, 2, 3, 17);

foreach($a as $v) {
   print "Valor actual de \$a: $v.\n";
}

/* foreach ejemplo 2: valor (con clave impresa para ilustrar) */
$a = array(1, 2, 3, 17);

$i = 0; /* s&oacute;lo para prop&oacute;sitos demostrativos */

foreach($a as $v) {
   print "\$a[$i] => $v.\n";
   $i++;
}

/* foreach ejemplo 3: clave y valor */
$a = array(
   "uno" => 1,
   "dos" => 2,
   "tres" => 3,
   "diecisiete" => 17
);

foreach($a as $k => $v) {
   print "\$a[$k] => $v.\n";
}

/* foreach ejemplo 4: matriz multi-dimensional */

$a[0][0] = "a";
$a[0][1] = "b";
$a[1][0] = "y";
$a[1][1] = "z";

foreach($a as $v1) {
    foreach ($v1 as $v2) {
        print "$v2\n";
    }
}

/* foreach ejemplo 5: matriz din&aacute;mica */

foreach(array(1, 2, 3, 4, 5) as $v) {
    print "$v\n";
}
?>
```

---

break

break escapa de la estructuras de control iterante (bucle) actuales *for*, *while*, o *switch*.

break accepta un parámetro opcional, el cual determina cuantas estructuras de control hay que escapar.

```php
<?php
$arr = array ('one', 'two', 'three', 'four', 'stop', 'five');
while (list (, $val) = each ($arr)) {
    if ($val == 'stop') {
        break;    /* You could also write 'break 1;' here. */
    }
    echo "$val<br>\n";
}

/* Using the optional argument. */

$i = 0;
while (++$i) {
    switch ($i) {
    case 5:
        echo "At 5<br>\n";
        break 1;  /* Exit only the switch. */
    case 10:
        echo "Al 10; saliendo<br>\n";
        break 2;  /* Exit the switch and the while. */
    default:
        break;
    }
}
?>
```

---

continue

continue se usa dentro de la estructura del bucle para saltar el resto de la iteración actual del bucle y continuar la ejecución al comienzo de la siguiente iteración.

> **Nota:** Tener en cuenta que en PHP la declaración switch es considerada una estructura de bucle por *continue*.

continue accepta un parámetro opcional, el cual determina cuantos niveles (bluces) hay que saltar antes de continuar con la ejecución.

```php
<?php
while (list ($key, $value) = each ($arr)) {
    if (!($key % 2)) { // skip odd members
        continue;
    }
    do_something_odd ($value);
}

$i = 0;
while ($i++ < 5) {
    echo "Outer<br>\n";
    while (1) {
        echo "  Middle<br>\n";
        while (1) {
            echo "  Inner<br>\n";
            continue 3;
        }
        echo "Esto nunca se imprime.<br>\n";
    }
    echo "Y esto tampoco.<br>\n";
}
?>
```

---

switch

La sentencia *switch* es similar a una serie de sentencias IF en la misma expresión. En muchas ocasiones, se quiere comparar la misma variable (o expresión) con nuchos valores diferentes, y ejecutar una parte de código distinta dependiendo de a qué valor es igual. Para ello sirve la sentencia *switch*.

> **Nota:** Tener en cuenta que al contrario que otros lenguajes de programación, continue se aplica a switch y funciona de manera similar a *break*. Si teneis un switch dentro de un bucle y deseais continuar con el paso siguiente en el bucle externo, usar *continue 2*.

Los siguientes dos ejemplos son dos modos distintos de escribir la misma cosa, uno usa una serie de sentencias *if*, y el otro usa la sentencia *switch*:

```
<?php
if ($i == 0) {
    print "i equals 0";
} elseif ($i == 1) {
    print "i equals 1";
} elseif ($i == 2) {
    print "i equals 2";
}

switch ($i) {
    case 0:
        print "i equals 0";
        break;
    case 1:
        print "i equals 1";
        break;
    case 2:
        print "i equals 2";
        break;
}
?>
```

Es importante entender cómo se ejecuta la sentencia *switch* para evitar errores. La sentencia *switch* ejecuta línea por línea (realmente, sentencia a sentencia). Al comienzo, no se ejecuta código. Sólo cuando se encuentra una sentencia *case* con un valor que coincide con el valor de la expresión *switch* PHP comienza a ejecutar las sentencias. PHP continúa ejecutando las sentencias hasta el final del bloque *switch*, o la primera vez que vea una sentencia *break*. Si no se escribe una sentencia *break* al final de una lista de sentencias case, PHP seguirá ejecutando las sentencias del siguiente case. Por ejemplo:

```
<?php
 switch ($i) {
     case 0:
         print "i es igual a 0";
     case 1:
         print "i es igual a 1";
     case 2:
         print "i es igual a 2";
 }
?>
```

Aquí, si $i es igual a 0, ¡PHP ejecutaría todas las sentecias print! Si $i es igual a 1, PHP ejecutaría las últimas dos sentencias print y sólo si $i es igual a 2, se obtendría la conducta 'esperada' y solamente se mostraría 'i es igual a 2'. Así, es importante no olvidar las sentencias *break* (incluso aunque pueda querer evitar escribirlas intencionadamente en ciertas circunstancias).

En una sentencia *switch*, la condición se evalúa sólo una vez y el resultado se compara a cada sentencia *case*. En una sentencia *elseif*, la condición se evalúa otra vez. Si tu condición es más

complicada que una comparación simple y/o está en un bucle estrecho, un *switch* puede ser más rápido.

La lista de sentencias de un case puede también estar vacía, lo cual simplemente pasa el control a la lista de sentencias del siguiente case.

```
<?php
switch ($i) {
     case 0:
     case 1:
     case 2:
         print "i es menor que 3, pero no negativo";
         break;
     case 3:
         print "i es 3";
 }
?>
```

Un caso especial es el *default* case". Este "case" coincide con todo lo que no coincidan los otros case. Por ejemplo:

```
<?php
 switch ($i) {
     case 0:
         print "i es igual a 0";
         break;
     case 1:
         print "i es igual a 1";
         break;
     case 2:
         print "i es igual a 2";
         break;
     default:
         print "i no es igual a 0, 1 o 2";
 }
?>
```

La expresión *case* puede ser cualquier expresión que se evalúe a un tipo simple, es decir, números enteros o de punto flotante y cadenas de texto. No se pueden usar aquí ni arrays ni objetos a menos que se conviertan a un tipo simple.

La sintaxis alternativa para las estructuras de control está también soportada con switch. Para más información, ver Sintaxis alternativa para estructuras de control.

```
<?php
switch ($i):
     case 0:
         print "i es igual 0";
         break;
     case 1:
         print "i es igual a 1";
         break;
     case 2:
         print "i es igual a 2";
         break;
     default:
         print "i no es igual a 0, 1 o 2";
 endswitch;
?>
```

---

declare

La construcción *declare* es usada para definir directivas de ejecución para un bloque de código. La

sintaxis de *declare* es similar a la de las otras estructuras de control:

```
declare (directiva) sentencia
```

Directiva permite asignar el comportamiento del bloque *declare*. Actualmente una sola directiva es reconocida: la directiva *ticks* (Consultar más abajo la información sobre la directiva ticks)

La *sentencia* es lo que se ejecuta -- Como se ejecuta y que efectos secundarios tiene depende de la directiva definida en la *directiva*.

El constructor *declare* se puede usar tambien globalmente, afectando a todo el código que le sigue.

```
<?php
// Estos son lo mismo:

// se puede usar este:
declare(ticks=1) {
    // script completo aqui
}

// o este:
declare(ticks=1);
// script completo aqui
?>
```

Ticks

Un "tick" es un evento que ocurre por cada *N* sentencias de bajo nivel ejecutadas dentro del bloque *declare*. El valor de *N* es especificado por *ticks=N* como *directiva* dentro de *declare*.

El evento que ocurre en cada "tick" es especificado usando la función **register_tick_function()**. Ver el ejemplo más abajo para más detalles. Tener en cuenta que mas de un evento puede ocurrir por cada "tick"

Ejemplo 16-1. Perfilar una sección de código PHP

```php
<?php
// A function that records the time when it is called
function profile ($dump = FALSE)
{
    static $profile;

    // Return the times stored in profile, then erase it
    if ($dump) {
        $temp = $profile;
        unset ($profile);
        return ($temp);
    }

    $profile[] = microtime ();
}

// Set up a tick handler
register_tick_function("profile");

// Initialize the function before the declare block
profile ();

// Run a block of code, throw a tick every 2nd statement
declare (ticks=2) {
    for ($x = 1; $x < 50; ++$x) {
        echo similar_text (md5($x), md5($x*$x)), "<br />;";
    }
}

// Display the data stored in the profiler
print_r (profile (TRUE));
?>
```

Este ejemplo perfila el código PHP dentro del bloque 'declare', grabando la hora, una sentencia si y otra no, cuando fue ejecutada. Esta información puede ser usada para encontrar areas en donde el codigo es lento. Este proceso se puede implementar de diferentes maneras: usando "ticks" es más conveniente y facil de implementar.

"Ticks" es una manera muy buena de eliminar errores, implementando simples trabajos en paralelo, I/O en modo desatendido y otras tareas.

Ver también **register_tick_function()** y **unregister_tick_function()**.

---

return

Si se llama desde una función, **return()** termina inmediatamente la ejecución de la función y retorna su argumento como valor de la función. **return()** tambien terminará la ejecución de una sentencia **eval()**ó un script PHP.

Si el script actual ha sido incluido ó requerido con **include()** ó **require()**, el control es transferido al script que llamo al script incluido. Además, si el script actual fue incluido, el valor dado a **return()** será retornado como el valor de la llamada **include()**. Si **return()** es invocado desde el script principal, la ejecución terminara inmediatamente. Si el script actual fue incluido con las opciones de configuración auto_prepend_file ó auto_append_file, la ejecución terminara inmediatamente.

Para más información, consultar Retornando valores.

> **Nota:** Tener en cuenta que ya que **return()** es un constructor del lenguaje y no una función, los paréntesis alrededor de sus argumentos, son *solo* necesarios si el argumento contiene una expresion, no se suelen utilizar tan a menudo, cuando retornan una

variable.

---

require()

La sentencia **require()** incluye y evalua el archivo especificado.

require() incluye y evalua el archivo especificado. Información detallada de como esta inclusión funciona se puede encontrar en la documentación de la función **include()**.

require() y **include()** son idénticas en todos los aspectos excepto en el modo de actuar ante un error. **include()** produce un Warning mientras que **require()** produce un Error Fatal. En otras palabras, no dude en utilizar **require()** si quiere que un fichero no encontrado cuelgue el procesamiento de la página. **include()** no se comporta de esta manera, el script seguirá funcionando de todas maneras. Asegurarse que include_path este configurado bien.

Ejemplo 16-2. ejemplos básicos de require()

```
<?php

require 'prepend.php';

require $somefile;

require ('somefile.txt');

?>
```

consultar la documentación de **include()** para más ejemplos.

> **Nota:** Con anterioridad a PHP 4.0.2, se aplica lo siguiente: **require()** siempre intentará leer el fichero a incluir, incluso si la linea donde se encuentra **require()** nunca es ejecutada. Sin embargo, si la linea donde se encuentra **require()** no es ejecutada, tampoco lo hará el código incluido.

> **Nota:** Puesto que esto es una construcción del lenguaje y no una función, no puede ser llamado usando funciones variables

Aviso
Versiones de *PHP* para Windows anteriores a 4.3.0, no soportan el acceso remoto a archivos para esta función, no funcionará ni activando siquiera allow_url_fopen.

Ver también **include()**, **require_once()**, **include_once()**, **eval()**, **file()**, **readfile()**, **virtual()** y include_path.

---

include()

La sentencia **include()** incluye y evalúa el archivo especificado.

Esta documentación también se aplica a la función **require()**. **require()** y **include()** son idénticas en todos los aspectos excepto en el modo de actuar ante un error. **include()** produce un Warning mientras que **require()** produce un Error Fatal. En otras palabras, no dude en utilizar **require()** si quiere que un fichero no encontrado cuelgue el procesamiento de la página. **include()** no se comporta de esta manera, el script seguirá funcionando de todas maneras. Asegurarse que

include_path este configurado bien.

Cuando un fichero es incluido, el código que contiene hereda la variable scope de la linea en donde el include ocurre. Cualquier variable disponible en esa linea en el fichero desde donde se hace la inclusión estará disponible en el fichero incluido a partir de ese momento.

Ejemplo 16-3. Ejemplo básico de la funcióninclude()

```
vars.php
<?php

$color = 'green';
$fruit = 'apple';

?>

test.php
<?php

echo "A $color $fruit"; // A

include 'vars.php';

echo "A $color $fruit"; // A green apple

?>
```

Si la inclusión ocurre dentro de una función en el fichero donde se incluye, todo el código contenido en el fichero incluido se comportará como si hubiese sido definido dentro de esta función.

Ejemplo 16-4. Incluyendo desde funciones

```
<?php

function foo()
{
    global $color;

    include 'vars.php';

    echo "A $color $fruit";
}

/* vars.php is in the scope of foo() so     *
 * $fruit is NOT available outside of this  *
 * scope.  $color is because we declared it *
 * as global.                               */

foo();                    // A green apple
echo "A $color $fruit";   // A green

?>
```

Cuando un fichero es incluido, el intérprete sale del modo PHP y entra en modo HTML al principio del archivo referenciado, y vuelve de nuevo al modo PHP al final. Por esta razón, cualquier código dentro del archivo referenciado que debiera ser ejecutado como código PHP debe ser encerrado dentro de etiquetas válidas de comienzo y fin de PHP.

Si "URL fopen wrappers" esta activada en PHP (como está en la configuración inicial), se puede especificar el fichero que se va a incluir usando una URL (via HTTP u otro mecanismo soportado, consultar Apéndice L) en vez de un fichero local. Si el servidor destino interpreta el fichero destino como código PHP, variables pueden ser mandadas al fichero incluido usando una cadena URL de petición, tal como se hace con HTTP GET. Esto no es lo mismo que incluir un fichero y que este fichero herede las variables del fichero padre; el script es ejecutado en el servidor remoto y el resultado es incluido en en script local.

Aviso
Versiones de *PHP* para Windows anteriores a 4.3.0, no soportan el acceso remoto a archivos para esta función, no funcionará ni activando siquiera allow_url_fopen.

Ejemplo 16-5. include() a traves de HTTP

```
<?php

/* This example assumes that www.example.com is configured to parse .php
 * files and not .txt files. Also, 'Works' here means that the variables
 * $foo and $bar are available within the included file.*/

// Won't work; file.txt wasn't handled by www.example.com as PHP
include 'http://www.example.com/file.txt?foo=1&bar=2';

// Won't work; looks for a file named 'file.php?foo=1&bar=2' on the
// local filesystem.
include 'file.php?foo=1&bar=2';

// Works.
include 'http://www.example.com/file.php?foo=1&bar=2';

$foo = 1;
$bar = 2;
include 'file.txt';  // Works.
include 'file.php';  // Works.

?>
```

Ver también Ficheros remotos, **fopen()** y **file()** para obtener información adicional.

Ya que **include()** y **require()** son contructores especiales del lenguaje, se deben de incluir dentro del bloque de una sentencia, si están dentro de un bloque condicional.

Ejemplo 16-6. include() y bloques condicionales

```
<?php

// This is WRONG and will not work as desired.
if ($condition)
    include $file;
else
    include $other;


// This is CORRECT.
if ($condition) {
    include $file;
} else {
    include $other;
}

?>
```

Es posible ejecutar una sentencia *return* dentro de un archivo incluido para terminar el procesado de ese archivo y volver al archivo de comandos que lo llamó. Tambien es posible retornar valores de ficheros incluidos. Se puede coger el valor de la llamada "include" como se haría con una función normal.

> **Nota:** En PHP3, *return* no puede aparecer dentro de un bloque a menos que sea un bloque de función, en el cual *return* se aplica a esa función y no al archivo completo.

Ejemplo 16-7. include() y return()

```
return.php
<?php

$var = 'PHP';

return $var;

?>

noreturn.php
<?php

$var = 'PHP';

?>

testreturns.php
<?php

$foo = include 'return.php';

echo $foo; // prints 'PHP'

$bar = include 'noreturn.php';

echo $bar; // prints 1

?>
```

$bar es igual a *1* porque la inclusión salio bien. Notar la diferencia entre los dos ejemplos anteriores. el primero usa **return()** dentro del fichero incluido y el segundo no. Otras maneras de incluir ficheros en variables es con **fopen()**, **file()** ó usando **include()**con Funciones de control de salida.

> **Nota:** Puesto que esto es una construcción del lenguaje y no una función, no puede ser llamado usando funciones variables

Ver también **require()**, **require_once()**, **include_once()**, **readfile()**, **virtual()**, y include_path.

---

require_once()

La función **require_once()** incluye y evalua el fichero especificado durante la ejecución del script. Se comporta de manera similar a **require()**, con la única diferencia que si el código ha sido ya incluido, no se volverá a incluir. Consultar la documentación de la función **require()** para obtener más información.

require_once() debería de usarse en casos en los que un mismo fichero puede ser incluido y evaluado más de una vez durante la ejecución de un script, y se quiere estar seguro que se incluye una sola vez para evitar problemas con redefiniciones de funciones, valores de funciones, etc.

Para consultar ejemplos que usen **require_once()** y **include_once()**, ver el código de PEAR incluido con las últimas versiones de PHP.

> **Nota:** **require_once()** fue incorporado en PHP 4.0.1pl2

> **Nota:** El comportamiento de de **require_once()** y **include_once()** puede que no sea el esperado en sistemas operativos en los que mayúsculas y minúsculas se traten igual (como en Windows)
> **Ejemplo 16-8. Con require_once() no importan las mayúsculas y minúsculas en Windows**

```
<?php
require_once("a.php"); // this will include a.php
require_once("A.php"); // this will include a.php again on Windows!
?>
```

Aviso
Versiones de *PHP* para Windows anteriores a 4.3.0, no soportan el acceso remoto a archivos para esta función, no funcionará ni activando siquiera allow_url_fopen.

Ver también **require()**, **include()**, **include_once()**, **get_required_files()**, **get_included_files()**, **readfile()**, y **virtual()**.

---

include_once()

La función **include_once()** incluye y evalua el fichero especificado durante la ejecución del script. Se comporta de manera similar a **include()**, con la única diferencia que si el código ha sido ya incluido, no se volverá a incluir.

include_once() debería de usarse en casos en los que, un mismo fichero puede ser incluido y evaluado más de una vez durante la ejecución de un script, y se quiere estar seguro que se incluye una sola vez para evitar problemas con redefiniciones de funciones, valores de funciones, etc.

Para consultar ejemplos que usen **include_once()** e **require_once()**, ver el código de PEAR incluido con las últimas versiones de PHP.

> **Nota:** **include_once()** fue incorporado en PHP 4.0.1pl2
>
> **Nota:** El comportamiento de de **include_once()** y **require_once()** puede que no sea el esperado en sistemas operativos en los que mayúsculas y minúsculas se traten igual (como en Windows)
> **Ejemplo 16-9. Con include_once() no importan las mayúsculas y minúsculas en Windows**
>
> ```
> <?php
> include_once("a.php"); // this will include a.php
> include_once("A.php"); // this will include a.php again on Windows!
> ?>
> ```

Aviso
Versiones de *PHP* para Windows anteriores a 4.3.0, no soportan el acceso remoto a archivos para esta función, no funcionará ni activando siquiera allow_url_fopen.

Ver también **include()**, **require()**, **require_once()**, **get_required_files()**, **get_included_files()**, **readfile()**, y **virtual()**.

Capítulo 17. Funciones

Funciones definidas por el usuario

Una función se puede definir con la siguiente sintaxis:

Ejemplo 17-1. Psuedo código para demostrar el uso de funciones

```
<?php
function foo ($arg_1, $arg_2, ..., $arg_n)
{
    echo "Funci&oacute;n de ejemplo.\n";
    return $retval;
}
?>
```

Cualquier instrucción válida de PHP puede aparecer en el cuerpo de la función, incluso otras funiones y definiciones de clases.

En PHP3, las funciones deben definirse antes de que se referencien. En PHP4 no existe tal requerimiento.*Excepto* cuando una función es definida condicionalmente como en los ejemplos siguientes.

Cuando una función es definida condicionalmente como se puede ver en estos dos ejemplos, su definición debe ser procesada *antes* que sea llamada.

Ejemplo 17-2. Funciones Condicionales

```
<?php

$makefoo = true;

/* We can't call foo() from here
   since it doesn't exist yet,
   but we can call bar() */

bar();

if ($makefoo) {
  function foo ()
  {
    echo "I don't exist until program execution reaches me.\n";
  }
}

/* Now we can safely call foo()
   since $makefoo evaluated to true */

if ($makefoo) foo();

function bar()
{
  echo "I exist immediately upon program start.\n";
}

?>
```

Ejemplo 17-3. Funciones dentro de funciones

```
<?php
function foo()
{
  function bar()
  {
    echo "I don't exist until foo() is called.\n";
  }
}

/* We can't call bar() yet
   since it doesn't exist. */

foo();

/* Now we can call bar(),
   foo()'s processesing has
   made it accessible. */

bar();

?>
```

PHP no soporta la redefinición de funciones previamente declaradas.

> **Nota:** Los nombres de funciones se pueden llamar con mayusculas o minusculas, aunque es una buena costumbre el llamar a las funciones tal y como aparecen en su definición.

PHP3 no soporta un número variable de parámetros, aunque sí soporta parámetros por defecto (ver Valores por defecto de de los parámetros para más información). PHP4 soporta ambos: ver Listas de longitud variable de parámetros y las referencias de las funciones **func_num_args()**, **func_get_arg()**, y **func_get_args()** para más información.

Parámetros de las funciones

La información puede suministrarse a las funciones mediante la lista de parámetros, una lista de variables y/o constantes separadas por comas.

PHP soporta pasar parámetros por valor (el comportamiento por defecto), por referencia, y parámetros por defecto. Listas de longitud variable de parámetros sólo están soportadas en PHP4 y posteriores; ver Listas de longitud variable de parámetros y la referencia de las funciones **func_num_args()**, **func_get_arg()**, y **func_get_args()** para más información. Un efecto similar puede conseguirse en PHP3 pasando un array de parámetros a la función:

Ejemplo 17-4. Pasando matrices a funciones

```
<?php
function takes_array($input)
{
    echo "$input[0] + $input[1] = ", $input[0]+$input[1];
}
?>
```

Pasar parámetros por referencia

Por defecto, los parámetros de una función se pasan por valor (de manera que si cambias el valor del argumento dentro de la función, no se ve modificado fuera de ella). Si deseas permitir a una función modificar sus parámetros, debes pasarlos por referencia.

Si quieres que un parámetro de una función siempre se pase por referencia, puedes anteponer un ampersand (&) al nombre del parámetro en la definición de la función:

Ejemplo 17-5. Pasando parámetros de funciones por referencia

```
<?php
function add_some_extra(&$string)
{
    $string .= ' y algo m&aacute;s.';
}
$str = 'Esto es una cadena, ';
add_some_extra($str);
echo $str;    // Saca 'Esto es una cadena, y algo m&aacute;s.'
?>
```

Parámetros por defecto

Una función puede definir valores por defecto para los parámetros escalares estilo C++:

Ejemplo 17-6. Uso de parámetros por defecto en funciones

```
<?php
function makecoffee ($type = "cappucino")
{
    return "Hacer una taza de $type.\n";
}
echo makecoffee ();
echo makecoffee ("espresso");
?>
```

La salida del fragmento anterior es:

```
Hacer una taza de cappucino.
Hacer una taza de espresso.
```

El valor por defecto tiene que ser una expresión constante, y no una variable, miembro de una clase ó llamada a una función.

Destacar que cuando se usan parámetros por defecto, estos tienen que estar a la derecha de cualquier parámetro sin valor por defecto; de otra manera las cosas no funcionarán de la forma esperada. Considera el siguiente fragmento de código:

Ejemplo 17-7. Uso incorrecto de parámetros por defecto en funciones

```
<?php
function makeyogurt ($type = "acidophilus", $flavour)
{
    return "Haciendo un bol de $type $flavour.\n";
}

echo makeyogurt ("mora");   // No funcionar&aacute; de la manera
esperada
?>
```

La salida del ejemplo anterior es:

```
Warning: Missing argument 2 in call to makeyogurt() in
/usr/local/etc/httpd/htdocs/php3test/functest.html on line 41
Haciendo un bol de mora.
```

Y ahora, compáralo con:

Ejemplo 17-8. Uso correcto de parámetros por defecto en funciones

```
<?php
function makeyogurt ($flavour, $type = "acidophilus")
{
    return "Haciendo un bol de $type $flavour.\n";
}

echo makeyogurt ("mora");   // funciona como se esperaba
?>
```

La salida de este ejemplo es:

```
Haciendo un bol de acidophilus mora.
```

Lista de longitud variable de parámetros

PHP4 soporta las listas de longitud variable de parámetros en las funciones definidas por el usuario. Es realmente fácil, usando las funciones **func_num_args()**, **func_get_arg()**, y **func_get_args()**.

No necesita de ninguna sintaxis especial, y las listas de parámetros pueden ser escritas en la llamada a la función y se comportarán de la manera esperada.

Devolviendo valores

Los valores se retornan usando la instrucción opcional return. Puede devolverse cualquier tipo de valor, incluyendo listas y objetos.

Ejemplo 17-9. Us0 de return()

```
<?php
function square ($num)
{
    return $num * $num;
}
echo square (4);   // saca '16'.
?>
```

No puedes devolver múltiples valores desde una función, pero un efecto similar se puede conseguir devolviendo una lista.

Ejemplo 17-10. Retornando una matriz para obtener múltiples valores

```
<?php
function small_numbers()
{
    return array (0, 1, 2);
}
list ($zero, $one, $two) = small_numbers();
?>
```

Para retornar una referencia desde una función, se tiene que usar el operador de referencias & tanto en la declaración de la función como en la asignación del valor de retorno a una variable;

Ejemplo 17-11. Retornando una referencia desde una función

```
<?php
function &returns_reference()
{
    return $someref;
}

$newref =& returns_reference();
?>
```

Para más información sobre referencias,consultar Explicando Referencias.

Funciones variables

PHP soporta el concepto de funciones variable, esto significa que si una variable tiene unos paréntesis añadidos al final, PHP buscará una función con el mismo nombre que la evaluación de la variable, e intentará ejecutarla. Entre otras cosas, esto te permite implementar retrollamadas (callbacks), tablas de funciones y demás.

Las funciones varibles no funcionarán con construcciones del lenguaje, tal como **echo()**, **print()**, **unset()**, **isset()**, **empty()**, **include()**, **require()** y derivados. Se necesitará usar una función propia para utilizar cualquiera de estos constructores como funciones variables.

Ejemplo 17-12. Ejemplo de función variable

```
<?php
function foo()
{
    echo "In foo()<br>\n";
}

function bar($arg = '')
{
    echo "In bar(); argument was '$arg'.<br>\n";
}

// This is a wrapper function around echo
function echoit($string)
{
    echo $string;
}

$func = 'foo';
$func();        // This calls foo()

$func = 'bar';
$func('test');  // This calls bar()

$func = 'echoit';
$func('test');  // This calls echoit()
?>
```

También se puede llamar a un metodo de un objeto usando la característica variable de las funciones.

Ejemplo 17-13. Ejemplo sobre el metodo variable

```php
<?php
class Foo
{
    function Var()
    {
        $name = 'Bar';
        $this->$name(); // This calls the Bar() method
    }

    function Bar()
    {
        echo "This is Bar";
    }
}

$foo = new Foo();
$funcname = "Var";
$foo->$funcname();  // This calls $foo->Var()

?>
```

Ver también **call_user_func()**, variable variables y **function_exists()**.

Funciones internas (incorporadas)

PHP tiene incorporadas muchas funciones y construcciones. Existen también funciones que requieren extensiones específicas de PHP para que no fallen con un error fatal del tipo "undefined function". Por ejemplo, para usar funciones image, tal como **imagecreatetruecolor()**, se necesita compilar PHP con soporte para GD. O para usar **mysql_connect()** se necesita compilar PHP con soporte para MySQL. Existen muchas funciones en el núcleo de PHP que se incluyen en cada version de PHP, como las funciones string y variable. Una llamada a la función **phpinfo()** ó **get_loaded_extensions()** mostrará que extensiones están cargadas en tu versión de PHP. Tener tambien en cuenta que muchas extensiones se encuentran activadas por defecto y que el manual de PHP se encuentra dividido en partes, según estas extensiones. Vea los capítulos configuración, instalación y los capitulos sobre cada extensión, para obtener información sobre como configurar vuestro PHP

La explicación de como leer e intrerpretar un prototipo de función se encuentra en la sección del manual titulada como leer la definición de una función. Es importante entender que devuelve una función ó si la función trabaja directamente en el valor entregado a la misma. Por ejemplo, **str_replace()** devuelve una cadena modificada mientras que **usort()** trabaja directamente en el valor entregado a la misma. Cada página del manual contiene información específica sobre las diferentes funciones existentes, parametros que utilizan, valores devueltos, cambios de comportamiento, etc. Es importante conocer estas diferencias para poder escribir correctamente código PHP.

Vea también **function_exists()**, referencias de funciones, **get_extension_funcs()** y **dl()**.

Capítulo 18. Clases y Objetos (PHP 4)

class

Una clase es una colección de variables y funciones que trabajan con éstas variables. Una clase es definida usando la siguiente sintaxis:

```
<?php
class Carrito {
    var $items;  // Items en nuestro carrito de compras

    // Agregar $num articulos de $artnr al carrito

    function agregar_item($artnr, $num) {
        $this->items[$artnr] += $num;
    }

    // Tomar $num articulos de $artnr del carrito

    function retirar_item($artnr, $num) {
        if ($this->items[$artnr] > $num) {
            $this->items[$artnr] -= $num;
            return true;
        } else {
            return false;
        }
    }
}
?>
```

Esto define una clase llamada Carrito que consiste de una matriz asociativa de artículos en el carrito y dos funciones para agregar y eliminar elementos del carrito.

Aviso

NO es posible separar la definición de una clase en varios archivos, o bloques PHP diferentes. Lo siguiente no funciona:

```
<?php
class prueba {
?>
<?php
    function prueba() {
        print 'Bien';
    }
}
?>
```

Las siguientes notas de precaución son válidas para PHP 4.

Atención

El nombre *stdClass* es usado internamente por Zend y es reservado. No puede tener una clase con el nombre *stdClass* en PHP.

Atención

Los nombres de función *__sleep* y *__wakeup* son mágicos en las clases PHP. No puede tener funciones con éstos nombres en cualquiera de sus clases a menos que desee usar la funcionalidad mágica asociada con ellas. Vea más información a continuación.

Atención

PHP reserva todos los nombres de función que comienzan con __ como mágicos. Se recomienda que no use nombres de función con __ en PHP a menos que desee usar alguna funcionalidad mágica documentada.

En PHP 4, sólo se permiten inicializadores constantes para variables *var*. Para inicializar variables con valores no-constantes, necesita una función de inicialización que sea llamada automáticamente cuando un objeto es construido a partir de la clase. Tal función es llamada constructora (vea más información a continuación).

```php
<?php
class Carrito {
    /* Ninguna de estas expresiones funciona en PHP 4. */
    var $fecha_hoy = date("Y-m-d");
    var $nombre = $primer_nombre;
    var $duenyo = 'Fred ' . 'Jones';
    /* Aunque, las matrices que contienen valores constantes funcionan */
    var $items = array("VCR", "TV");
}

/* Asi es como debe declararse. */
class Carrito {
    var $fecha_hoy;
    var $nombre;
    var $duenyo;
    var $items = array("VCR", "TV");

    function Carrito() {
        $this->fecha_hoy = date("Y-m-d");
        $this->nombre = $GLOBALS['primer_nombre'];
        /* etc. . . */
    }
}
?>
```

Las clases son tipos, es decir, son planos usados para variables reales. Necesita crear una variable del tipo deseado con el operador *new*.

```php
<?php
$carrito = new Carrito;
$carrito->agregar_item("10", 1);

$otro_carrito = new Carrito;
$otro_carrito->agregar_item("0815", 3);
?>
```

Esto crea los objetos *$carrito* y *$otro_carrito*, ambos de la clase Carrito. La función agregar_item() del objeto *$carrito* es llamada para agregar 1 item del artículo número 10 al *$carrito*. Se agregan 3 items del artículo número 0815 al *$otro_carrito*.

Ambos, *$carrito* y *$otro_carrito*, tienen funciones agregar_item(), retirar_item() y una variable items. Estas son variables y funciones diferentes. Puede pensar sobre los objetos como algo similar a las directorios en un sistema de archivos. En un sistema de archivos es posible tener dos archivos `LEAME.TXT` diferentes, siempre y cuando estén en directorios diferentes. Tal y como con los directorios, en donde es necesario escribir las rutas de nombres completas para llegar a cada archivo a partir del directorio del nivel superior, es necesario especificar el nombre completo de la función que desea llamar; en términos de PHP, el directorio de nivel superior sería el espacio de nombres global, y el separador de ruta sería ->. De tal modo que los nombres *$carrito->items* y *$otro_carrito->items* hacen referencia a dos variables diferentes. Note que la variable se llama *$carrito->items*, no *$carrito->$items*, es decir, un nombre de variable en PHP solo tiene un único signo de dólar.

```php
<?php
// correcto, un solo $
$carrito->items = array("10" => 1);

// invalido, ya que $carrito->$items se convierte en $carrito->""
$carrito->$items = array("10" => 1);

// correcto, pero puede o no ser lo que se busca:
// $carrito->$mivar se convierte en $carrito->items
$mivar = 'items';
$carrito->$mivar = array("10" => 1);
?>
```

Al interior de una definición de clase, no se conoce el nombre bajo el que el objeto será accesible en

su programa: en el momento en que la clase Carrito fue escrita, no se conocía que el objeto se llamaría *$carrito* u *$otro_carrito* más adelante. Por lo tanto, no es posible escribir *$carrito->items* al interior de la clase Carrito. En su lugar, para poder acceder a sus propias funciones y variables desde el interior de una clase, es posible usar la pseudo-variable *$this*, la cual puede leerse como 'mi propio' o 'el objeto actual'. Por lo tanto, '*$this->items[$num_art] += $num*' puede leerse como 'agregar *$num* al contador *$num_art* de mi propia matriz de items', o 'agregar *$num* al contador *$num_art* de la matriz de items al interior del objeto actual'.

> **Nota:** Existen algunas funciones interesantes que manejan clases y objetos. Puede que quiera echar un vistazo a las Funciones de Clase/Objeto.

---

extends

Con frecuencia es necesario tener clases con variables y funciones similares a otra clase existente. De hecho, es una buena práctica definir una clase genérica que pueda ser usada en todos sus proyectos y adaptar ésta clase para las necesidades de cada uno de sus proyectos específicos. Para facilitar esto, las clases pueden ser extensiones de otras clases. La clase extendida o derivada tiene todas las variables y funciones de la clase base (esto es llamado 'herencia', a pesar del hecho de que nadie muera) y lo que se agregue en la definición extendida. No es posible substraer de una clase, es decir, remover la definición de cualquier función o variable existente. Una clase extendida siempre depende de una clase base única, lo que quiere decir que no se soporta herencia múltiple. Las clases son extendidas usando la palabra clave 'extends'.

```
<?php
class Carrito_Con_Nombre extends Carrito {
    var $duenyo;

    function definir_duenyo ($nombre) {
        $this->duenyo = $nombre;
    }
}
?>
```

Esto define una clase Carrito_Con_Nombre que tiene todas las variables y funciones de Carrito, más una variable adicional *$duenyo* y una función adicional definir_duenyo(). Se crea un carrito con nombre en la forma usual y ahora es posible definir y obtener el dueño del carrito. Aun es posible usar las funciones normales de un carrito sobre los carritos con nombre:

```
<?php
$carrito_n = new Carrito_Con_Nombre;  // Crear un carrito con nombre
$carrito_n->definir_duenyo("kris");   // Nombrar el carrito
print $carrito_n->duenyo;             // imprimir el nombre del duenyo
$carrito_n->agregar_item("10", 1);    // (funcionalidad heredada de carrito)
?>
```

Esto también se conoce como una relación "padre-hijo". Se crea una clase, padre, y se usa *extends* para crear una clase *basada* en la clase padre: la clase hija. Es posible incluso usar esta nueva clase hija y crear otra clase basada en esta clase hija.

> **Nota:** ¡Las clases deben ser definidas antes de ser usadas! Si desea que la clase *Carrito_Con_Nombre* extienda la clase *Carrito*, tendrá que definir la clase *Carrito* primero. Si desea crear otra clase llamada *Carrito_amarillo_con_nombre* basada en la clase *Carrito_Con_Nombre*, debe definir *Carrito_Con_Nombre* primero. Resumiendo: el orden en que se definen las clases es importante.

Constructores

Atención
En PHP 3 y PHP 4 los constructores se comportan de forma diferente. La semántica de PHP 4 se prefiere considerablemente.

Los constructores son funciones en una clase que son llamadas automáticamente cuando se crea una nueva instancia de una clase con *new*. En PHP 3, una función se convierte en constructor si tiene el mismo nombre que la clase. En PHP 4, una función se convierte en constructor cuando tiene el mismo nombre que la clase en donde es definida - la diferencia es sutil, pero crucial (vea más adelante).

```
<?php
// Funciona en PHP 3 y PHP 4.
class Auto_Carrito extends Carrito {
    function Auto_Carrito() {
        $this->agregar_item("10", 1);
    }
}
?>
```

Esto define una clase Auto_Carrito que es un Carrito más un constructor que inicializa el carrito con un item del número de artículo "10" cada vez que un nuevo Auto_Carrito se crea con "new". Los constructores pueden recibir argumentos y tales argumentos pueden ser opcionales, lo que los hace mucho más útiles. Para poder usar aun la clase sin parámetros, todos los parámetros deben ser opcionales, al proveer valores predeterminados.

```
<?php
// Funciona en PHP 3 y PHP 4.
class Constructor_Carrito extends Carrito {
    function Constructor_Carrito($item = "10", $num = 1) {
        $this->agregar_item ($item, $num);
    }
}

// Comprar lo mismo de antes.

$carrito_predeterminado = new Constructor_Carrito;

// Comprar esta vez en serio...

$carrito_diferente = new Constructor_Carrito("20", 17);
?>
```

También puede usar el operador @ para *callar* los errores que ocurren en el constructor, p.ej. *@new*.

Atención
En PHP 3, las clases derivadas y los constructores tienen un número de limitaciones. Lo siguientes ejemplos deben ser leídos cuidadosamente para entender esas limitaciones.

```
<?php
class A {
    function A() {
      echo "Soy el constructor de A.<br />\n";
    }
}

class B extends A {
    function C() {
        echo "Soy una funci&oacute;n regular.<br />\n";
    }
}

// ne se esta llamando un constructor en PHP 3.
$b = new B;
?>
```

En PHP 3, no se está llamando un constructor en el ejemplo anterior. La regla en PHP 3 es: 'Un constructor es una función del mismo nombre que la clase.'. El nombre de la clase es B, y no hay una función llamada B() en la clase B. Nada ocurre.

Esto se corrige en PHP 4 al introducir otra regla: Si una clase no tiene constructor, el constructor de la clase base es llamado, si existe. El ejemplo anterior habría impreso 'Soy el constructor de A.
' en PHP 4.

```
<?php
class A
{
    function A()
    {
        echo "Soy el constructor de A.<br />\n";
    }

    function B()
    {
        echo "Soy una funci&oacute;n regular llamada B en la clase A.<br />\n";
        echo "No soy un constructor en A.<br />\n";
    }
}

class B extends A
{
    function C()
    {
        echo "Soy una funci&oacute;n regular.<br />\n";
    }
}

// Esto llama a B() como un constructor.
$b = new B;
?>
```

En PHP 3, la función B() en la clase A se convertirá de repente en un constructor en la clase B, aun cuando nunca fue esa la intención. La regla en PHP 3 es: 'Un constructor es una función con el mismo nombre de la clase.'. A PHP 3 no le importa si la función está siendo definida en la clase B, o si ha sido heredada.

Esto se corrige en PHP 4 modificando la regla a: 'Un constructor es una función del mismo nombre de la clase en la que se define.'. Por lo tanto, en PHP4, la clase B no tendría una función constructora propia y el constructor de la clase base hubiera sido llamado, imprimiendo 'Soy el constructor de A.
'.

Atención

Ni PHP 3 o PHP 4 llaman constructores de la clase base automáticamente desde un constructor de una clase derivada. Es su responsabilidad propagar la llamada a constructores más arriba en la jerarquía cuando sea apropiado.

> **Nota:** No hay destructores en PHP 3 o PHP 4. Puede usar **register_shutdown_function ()** en su lugar para emular la mayoría de efectos de los destructores.

Los destructores son funciones que son llamadas automáticamente cuando un objeto es destruido, ya sea con **unset()** o simplemente al finalizarse su contexto. No hay destructores en PHP.

::

Atención
Lo siguiente es válido únicamente para PHP 4 y versiones posteriores.

Algunas veces es útil referirse a funciones y variables en clases base o referirse a funciones en clases que no tienen aun alguna instancia. El operador :: es usado en tales casos.

```
<?php
class A {
    function ejemplo() {
        echo "Soy la funci&oacute;n original A::ejemplo().<br />\n";
    }
}

class B extends A {
    function ejemplo() {
        echo "Soy la funci&oacute;n redefinida B::ejemplo().<br />\n";
        A::ejemplo();
    }
}

// no hay un objeto de la clase A.
// esto imprime
//   Soy la funcion original A::ejemplo().<br />
A::ejemplo();

// crear un objeto de clase B.
$b = new B;

// esto imprime
//   Soy la funcion redefinida B::ejemplo().<br />
//   Soy la funcion original A::ejemplo().<br />
$b->ejemplo();
?>
```

El ejemplo anterior llama la función ejemplo() en la clase A, pero no hay un objeto de la clase A, así que no podemos escribir $a->ejemplo() o algo semejante. En su lugar llamamos ejemplo() como una 'función de clase', es decir, una función de la clase misma, no de un objeto de tal clase.

Existen funciones de clase, pero no existen variables de clase. De hecho, no hay un objeto en absoluto al momento de la llamada. Por lo tanto, una función de clase no puede usar variables del objeto (pero puede usar variables locales y globales), y no puede usar *$this*.

En el ejemplo anterior, la clase B redefine la función ejemplo(). La definición original en la clase A es cubierta y ya no se encuentra disponible, a menos que haga referencia específicamente a la implementación de ejemplo() en la clase A usando el operador ::. Escriba A::ejemplo() para hacerlo (en realidad, debería usar parent::ejemplo(), como se muestra en la siguiente sección).

En este contexto, hay un objeto actual y puede tener variables de objeto. Por lo tanto, cuando se usa

DESDE EL INTERIOR de una función de objeto, es posible usar *$this* y variables de objeto.

---

parent

Es posible que se encuentre escribiendo código que hace referencia a variables y funciones de las clases base. Esto es particularmente cierto si su clase derivada es una refinación o especialización del código en su clase base.

En lugar de usar el nombre literal de la clase base en su código, debería usar el nombre especial *parent*, el cual hace referencia al nombre de su clase base tal y como se entrega en la declaración *extends* de su clase. Al hacer esto, evita usar el nombre de su clase base en más de un lugar. Llegado el caso de que su árbol de jerarquía cambie durante la implementación, el cambio se puede efectuar con facilidad simplemente modificando la declaración *extends* de su clase.

```
<?php
class A {
    function ejemplo() {
        echo "Soy A::ejemplo() y ofrezco funcionalidad b&aacute;sica.<br />\n";
    }
}

class B extends A {
    function ejemplo() {
        echo "Soy B::ejemplo() y ofrezco funcionalidad adicional.<br />\n";
        parent::ejemplo();
    }
}

$b = new B;

// Esto hace la llamada a B::ejemplo(), la cual llama a su vez a A::ejemplo().
$b->ejemplo();
?>
```

Seriación de objetos, objetos en sesiones

> **Nota:** En PHP 3, los objetos perdían su asociación de clase a lo largo del proceso de seriación y decodificación. La variable resultante es de tipo objeto, pero no tiene clase ni métodos, de modo que es inútil (se ha convertido en algo como una matriz con una sintaxis curiosa).

Atención
La siguiente información es válida para PHP 4 únicamente.

serialize() devuelve una cadena que contiene una representación tipo secuencia-de-bytes de cualquier valor que pueda ser almacenado en PHP. **unserialize()** puede causar que éstas cadenas recreen los valores de variable originales. Usando el método de seriación para guardar un objeto guardará todas las variables en un objeto. Las funciones en un objeto no se guardarán, sólo el nombre de la clase.

Para poder usar **unserialize()** con un objeto, la clase de ese objeto necesita ser definida. Es decir, si tiene un objeto *$a* de la clase A en pagina1.php y codifica ésta variable, obtendrá una cadena que hace referencia a la clase A y contiene todos los valores de variables contenidas en *$a*. Si desea tener la capacidad de revertir el proceso de seriación en pagina2.php, recreando *$a* de la clase A, la definición de la clase A debe estar presente en pagina2.php. Esto puede conseguirse por ejemplo

almacenando la definición de la clase A en un archivo de inclusión e incluyéndolo tanto en pagina1.php como en pagina2.php.

```
<?php
// clase_a.inc:

  class A {
      var $uno = 1;

      function mostrar_uno() {
          echo $this->uno;
      }
  }

// pagina1.php:

  include("clase_a.inc");

  $a = new A;
  $s = serialize($a);
  // almacenar $s en alguna parte en donde pagina2.php lo pueda encontrar.
  $fp = fopen("almacenamiento", "w");
  fwrite($fp, $s);
  fclose($fp);

// pagina2.php:

  // esto es necesaria para revertir la seriacion apropiadamente.
  include("clase_a.inc");

  $s = implode("", @file("almacenamiento"));
  $a = unserialize($s);

  // ahora use la funcion mostrar_uno() del objeto $a.
  $a->mostrar_uno();
?>
```

Si está usando sesiones y usa **session_register()** para registrar objetos, éstos objetos son seriados automáticamente al final de cada página PHP, y son decodificados de vuelta automáticamente en cada una de las siguientes páginas. Esto quiere decir, básicamente, que tales objetos pueden aparecer en cualquiera de sus páginas una vez hacen parte de su sesión.

Es bastante recomendable que incluya las definiciones de clase de todos esos objetos registrados en todas sus páginas, incluso si no va a usar realmente éstas clases en todas sus páginas. Si no lo hace y un objeto está siendo decodificado sin que su definición de clase esté presente, perderá su asociación de clase y se convertirá en un objeto de la clase *stdClass* sin ninguna función disponible, es decir, se hará prácticamente inútil.

De modo que si en el ejemplo anterior *$a* se hacía parte de una sesión ejecutando *session_register ("a")*, debería incluírse el archivo *clase_a.inc* en todas sus páginas, no sólo en pagina1.php y pagina2.php.

Las funciones mágicas *__sleep* y *__wakeup*

serialize() revisa si su clase tiene una función con el nombre mágico *__sleep*. De ser así, esa función es ejecutada antes de cualquier intento de seriación. Puede limpiar el objeto y su intención es que devuelva una matriz con los nombres de todas las variables de ese objeto que deberían ser seriadas.

El uso planeado para *__sleep* es cerrar todas las conexiones de bases de datos que pueda tener el

objeto, aplicando datos pendientes o realizando tareas similares de limpieza. Asimismo, la función resulta útil si tiene objetos bastante grandes que no necesitan ser guardados en su totalidad.

De forma semejante, **unserialize()** revisa por la presencia de una función con el nombre mágico *__wakeup*. Si está presente, ésta función puede reconstruir cualquier recurso que el objeto pueda tener.

El uso planeado para *__wakeup* es reestablecer cualquier conexión con bases de datos que hayan podido perderse durante la seriación y realizar otras tareas de reinicialización.

Las referencias al interior del constructor

Crear referencias al interior del constructor puede llevar a resultados confusos. Esta sección tipo-tutorial le ayuda a evitar problemas.

```
<?php
class Foo {
    function Foo($nombre) {
        // crear una referencia al interior de la matriz global $refglobal
        global $refglobal;
        $refglobal[] = &$this;
        // definir el nombre al valor pasado
        $this->definirNombre($nombre);
        // e imprimirlo
        $this->imprimirNombre();
    }

    function imprimirNombre() {
        echo "<br />", $this->nombre;
    }

    function definirNombre($nombre) {
        $this->nombre = $nombre;
    }
}
?>
```

Revisemos si existe una diferencia entre *$bar1*, que ha sido creado usando el operador de copia = y *$bar2* que ha sido creado usando el operador de referencia =&...

```
<?php
$bar1 = new Foo('definido en el constructor');
$bar1->imprimirNombre();
$refglobal[0]->imprimirNombre();

/* salida:
definido en el constructor
definido en el constructor
definido en el constructor */

$bar2 =& new Foo('definido en el constructor');
$bar2->imprimirNombre();
$refglobal[1]->imprimirNombre();

/* salida:
definido en el constructor
definido en el constructor
definido en el constructor */
?>
```

Aparentemente no hay ninguna diferencia, pero en realidad hay una bastante importante: *$bar1* y *$refglobal[0]* _NO_ son referenciados, NO son la misma variable. Esto se debe a que "new" no

devuelve una referencia por defecto, en su lugar devuelve una copia.

> **Nota:** No existe una pérdida de rendimiento (ya que desde PHP 4 se usa el conteo de referencias) al devolver copias en lugar de referencias. Al contrario, con frecuencia es mejor trabajar simplemente con copias en lugar de referencias, ya que crear referencias toma cierto tiempo mientras que crear copias prácticamente no toma nada de tiempo (a menos que ninguna de ellas sea una matriz u objeto grande y una de ellas se modifique y luego las otras subsecuentemente, entonces sería buena idea usar referencias para modificarlas todas al mismo tiempo).

Para probar lo que se dice más arriba, veamos el siguiente código.

```
<?php
// ahora cambiaremos el nombre. que espera que pase?
// puede que espere que tanto $bar1 como $refglobal[0] cambien sus nombres...
$bar1->definirNombre('definido desde afuera');

// como se ha mencionado antes, ese no es el caso.
$bar1->imprimirNombre();
$refglobal[0]->imprimirNombre();

/* salida:
definido desde afuera
definido en el constructor */

// veamos que cambia entre $bar2 y $refglobal[1]
$bar2->definirNombre('definido desde afuera');

// por suerte, no solo son iguales, son la misma variable, de modo que
// $bar2->nombre y $refglobal[1]->nombre son el mismo tambien
$bar2->imprimirNombre();
$refglobal[1]->imprimirNombre();

/* salida:
definido desde afuera
definido desde afuera */
?>
```

Otro ejemplo final, intente entenderlo.

```php
<?php
class A {
    function A($i) {
        $this->valor = $i;
        // intente descubrir porque no necesitamos una referencia aqui
        $this->b = new B($this);
    }

    function crearRef() {
        $this->c = new B($this);
    }

    function echoValor() {
        echo "<br />","clase ",get_class($this),': ',$this->valor;
    }
}


class B {
    function B(&$a) {
        $this->a = &$a;
    }

    function echoValor() {
        echo "<br />","clase ",get_class($this),': ',$this->a->valor;
    }
}

// intente entender porque usar una simple copia produciria
// un resultado no deseado en la linea marcada con *
$a =& new A(10);
$a->crearRef();

$a->echoValor();
$a->b->echoValor();
$a->c->echoValor();

$a->valor = 11;

$a->echoValor();
$a->b->echoValor(); // *
$a->c->echoValor();

?>
```

Este ejemplo producirá la salida:

```
clase A: 10
clase B: 10
clase B: 10
clase A: 11
clase B: 11
clase B: 11
```

Comparación de objetos

En PHP 4, los objetos son comparados en una forma muy simple: Dos instancias de objeto son iguales si tienen los mismos atributos y valores, y son instancias de la misma clase. Reglas similares se aplican cuando se comparan dos objetos usando el operador de identidad (===).

Si ejecutáramos el código del siguiente ejemplo:
Ejemplo 18-1. Ejemplo de comparación de objetos en PHP 4

```
<?php
function bool_a_cadena($bool) {
    if ($bool === false) {
            return 'FALSE';
    } else {
            return 'TRUE';
    }
}

function compararObjetos(&$o1, &$o2) {
    echo 'o1 == o2 : '.bool_a_cadena($o1 == $o2)."\n";
    echo 'o1 != o2 : '.bool_a_cadena($o1 != $o2)."\n";
    echo 'o1 === o2 : '.bool_a_cadena($o1 === $o2)."\n";
    echo 'o1 !== o2 : '.bool_a_cadena($o1 !== $o2)."\n";
}

class Bandera {
    var $bandera;

    function Bandera($bandera=true) {
            $this->bandera = $bandera;
    }
}

class BanderaCambiante extends Bandera {

    function encender() {
        $this->bandera = true;
    }

    function apagar() {
        $this->bandera = false;
    }
}

$o = new Bandera();
$p = new Bandera(false);
$q = new Bandera();

$r = new BanderaCambiante();

echo "Comparar instancias creadas con los mismos par&aacute;metros\n";
compararObjetos($o, $q);

echo "\nComparar instancias creadas con par&aacute;metros diferentes\n";
compararObjetos($o, $p);

echo "\nComparar una instancia de una clase padre con una de una subclase\n";
compararObjetos($o, $r);
?>
```

Veremos:

```
Comparar instancias creadas con los mismos parámetros
o1 == o2 : TRUE
o1 != o2 : FALSE
o1 === o2 : TRUE
o1 !== o2 : FALSE

Comparar instancias creadas con parámetros diferentes
o1 == o2 : FALSE
o1 != o2 : TRUE
o1 === o2 : FALSE
o1 !== o2 : TRUE

Comparar una instancia de una clase padre con una de una subclase
o1 == o2 : FALSE
o1 != o2 : TRUE
o1 === o2 : FALSE
o1 !== o2 : TRUE
```

Que es la salida que podemos esperar dadas las reglas de comparación descritas anteriormente. Solo

las instancias con los mismos valores para sus atributos y de la misma clase son consideradas iguales e idénticas.

Incluso en los casos en donde tenemos composición de objetos, se aplican las mismas reglas de comparación. En el ejemplo siguiente creamos una clase contenedora que almacena una matriz asociativa de objetos **Bandera**.

Ejemplo 18-2. Comparación de objetos compuestos en PHP 4

```
<?php
class ConjuntoBanderas {
    var $conjunto;

    function ConjuntoBanderas($matrizBanderas = array()) {
        $this->conjunto = $matrizBanderas;
    }

    function agregarBandera($nombre, $bandera) {
        $this->conjunto[$nombre] = $bandera;
    }

    function eliminarBandera($nombre) {
        if (array_key_exists($nombre, $this->conjunto)) {
            unset($this->conjunto[$nombre]);
        }
    }
}


$u = new ConjuntoBanderas();
$u->agregarBandera('bandera1', $o);
$u->agregarBandera('bandera2', $p);
$v = new ConjuntoBanderas(array('bandera1'=>$q, 'bandera2'=>$p));
$w = new ConjuntoBanderas(array('bandera1'=>$q));

echo "\nObjetos compuestos u(o,p) y v(q,p)\n";
compararObjetos($u, $v);

echo "\nu(o,p) y w(q)\n";
compararObjetos($u, $w);
?>
```

Cosa que entrega la salida esperada:

```
Objetos compuestos u(o,p) y v(q,p)
o1 == o2 : TRUE
o1 != o2 : FALSE
o1 === o2 : TRUE
o1 !== o2 : FALSE

u(o,p) y w(q)
o1 == o2 : FALSE
o1 != o2 : TRUE
o1 === o2 : FALSE
o1 !== o2 : TRUE
```

Capítulo 19. Clases y Objetos (PHP 5)

Introducción

En PHP 5 hay un nuevo modelo de Objetos. El manejo de PHP de objetos ha sido reescrito por completo, permitiendo un mejor desempeño y mas características.

Las Bases

clase

Cada definición de case empieza con la palabra "class", seguida por un nombre de clase, el cual puede ser cualquier nombre que no esté en la lista de palabras reserved en PHP. Seguida por un par de llaves curvas, las cuales contienen la definición de los miembros de la case y los métodos. Una seudo variable *$this* está disponible cuando un método es llamado dentro del contexto de un objeto. *$this* es una referencia al objeto que se está usando (usualmente el objeto al que el método pertenece, pero puede ser otro objeto, si un método es llamado estáticamente desde el contexto de un objeto secundario). Est es ilustrado en el siguiente ejemplo:

```
<?php
class A
{
    function foo()
    {
        if (isset($this)) {
            echo '$this is defined (';
            echo get_class($this);
            echo ")\n";
        } else {
            echo "\$this is not defined.\n";
        }
    }
}

class B
{
    function bar()
    {
        A::foo();
    }
}

$a = new A();
$a->foo();
A::foo();
$b = new B();
$b->bar();
B::bar();
?>
```

El resultado del ejemplo seria:

```
$this is defined (a)
$this is not defined.
$this is defined (b)
$this is not defined.
```

Ejemplo 19-1. Definición simple de una clase

```
<?php
class SimpleClass
{
    // member declaration
    public $var = 'a default value';

    // method declaration
    public function displayVar() {
        echo $this->var;
    }
}
?>
```

Nuevo objeto

Para crear una instancia de un objeto, un nuevo objeto debe ser creado y asignado a una variable. Un objeto siempre será asignado cuando se crea un objeto nuevo a menos que el objeto tenga un constructor definido que arroje una excepción en caso de error.

Ejemplo 19-2. Creando una instancia

```
<?php
$instance = new SimpleClass()
?>
```

Cuando se asigna una instancia de un objeto previamente creado a una nueva variable, la nueva variable accesará la misma instancia que la del objeto a la que fue asignada. Este comportamiento es el mismo cuando se pasan instancias a una función. Una nueva instancia de un objeto previamente creado puede ser hecho clonandolo.

Ejemplo 19-3. Asignamiento de Objeto

```
<?php
$assigned   =  $instance;
$reference  =& $instance;

$instance->var = '$assigned will have this value';

$instance = null; // $instance and $reference become null

var_dump($instance);
var_dump($reference);
var_dump($assigned);
?>
```

El resultado del ejemplo seria:

```
NULL
NULL
object(SimpleClass)#1 (1) {
   ["var"]=>
     string(30) "$assigned will have this value"
}
```

Extendiendo objetos

Una clase puede heredar métodos y miembros de otra clase usando la palabra 'extends' en la declaración. No es posible extender de múltiples clases, una clase puede heredar solo de una clase base.

Los métodos de herencia y sus miembros pueden ser evitados, redeclarandolos con el mismo nombre con el que los definió la clase padre, a menos que la clase padre haya definido un método como final. Es posible accesar a los métodos o miembros redeclarados haciendo referencia a ellos con parent::

Ejemplo 19-4. Herencia de SimpleClass

```php
<?php
class ExtendClass extends SimpleClass
{
    // Redefine the parent method
    function displayVar()
    {
        echo "Extending class\n";
        parent::displayVar();
    }
}

$extended = new ExtendClass();
$extended->displayVar();
?>
```

El resultado del ejemplo seria:

```
Extending class
a default value
```

Auto carga de Objetos

Muchos desarrolladores que escriben aplicaciones con programación orientada a objetos crean un archivo fuente PHP por cada definición de clase. Una de las molestias más grandes es tener que escribir una larga lista de includes necesarios al principio de cada script (uno pra cada clase).

En PHP 5, esto ya no es necesario. Puede definir una función __autoload la cual es llamada automáticamente en caso de que intente usar una clase que no ha sido definida aún. Al llamar esta función la ejecución del script da una última oportunidad de cargar la clase antes de que PHP falle con un error.

> **Nota:** Las excepciones arrojadas en la función __autoload no pueden ser capturadas en el bloque catch y resultan en el despliegue de un error fatal.

Ejemplo 19-5. Ejemplo de auto carga

Este ejemplo intenta cargar las clases *MyClass1* y *MyClass2* de los archivos `MyClass1.php` y `MyClass2.php` respectivamente.

```php
<?php
function __autoload($class_name) {
    require_once $class_name . '.php';
}

$obj  = new MyClass1();
$obj2 = new MyClass2();
?>
```

Constructores y Destructores

Constructor

void **__construct** ([mixed args [, ...]])

PHP 5 permite a los desarrolladores declarar métodos constructores para las clases. Las clases que tiene un método constructor llaman a este método cada vez que se crea un nuevo objeto, para cualquier inicializació que el objeto puede necesitar antes de ser usado.

> **Nota:** Los constructores padres no son llamados implicitamente si la clase hijo define

un constructor. Para poder ejecutar el constructor de la clase padre, se debe hacer una llamada a **parent::__construct()** dentro del constructor de la clase hijo.

Ejemplo 19-6. Usando constructores unificados

```
<?php
class BaseClass {
   function __construct() {
       print "In BaseClass constructor\n";
   }
}

class SubClass extends BaseClass {
   function __construct() {
       parent::__construct();
       print "In SubClass constructor\n";
   }
}

$obj = new BaseClass();
$obj = new SubClass();
?>
```

Para tener compatibilidad con versiones anteriores, si PHP 5 no encuentra una función **__construct()** para una clase dada, buscará la forma de la función del constructor que se usaba anteriormente por el nombre de la clase. Efectivamente, esto significa que el único caso que puede tener compatibilidad es si la clase tiene un método llamado **__construct()** el cual fue usado para semánticas diferentes.

Destructores

void **__destruct** (void)

PHP 5 introduce un concepto de destructor similar a aquellos de otros lenguajes de programación orientada a objetos, tal como C++. El método destructor será llamado tan pronto como todas las referencias a un objeto en particular sean removidas o cuando el objeto sea explícitamente destruido.

Ejemplo 19-7. Ejemplo de Destructor

```
<?php
class MyDestructableClass {
   function __construct() {
       print "In constructor\n";
       $this->name = "MyDestructableClass";
   }

   function __destruct() {
       print "Destroying " . $this->name . "\n";
   }
}

$obj = new MyDestructableClass();
?>
```

Como los constructores, los destructores de la clase padre no serán llamados explícitamente por el compilador. Para ejecutar un destructor padre, se debe tener una llamada explícita a **parent::__destruct()** en el cuerpo del destructor.

Visibilidad

La visibilidad de una propiedad o método puede ser defionida al anteponerle a la declaración con las palabras reservadas: public, protected o private. Los elementos declarados con Public pueden ser accesados en todos lados. los Protected limitan el acceso a las clases heredadas (y alas clase que define el elemento). los Private limitan la visibilidad solo a la clase que lo definió.

Visibilidad de los miembros

Los miembros de la clase debe estar definidos con public, private o protected.

Ejemplo 19-8. Declaración de miembros

```
<?php
/**
 * Define MyClass
 */
class MyClass
{
    public $public = 'Public';
    protected $protected = 'Protected';
    private $private = 'Private';

    function printHello()
    {
        echo $this->public;
        echo $this->protected;
        echo $this->private;
    }
}

$obj = new MyClass();
echo $obj->public; // Works
echo $obj->protected; // Fatal Error
echo $obj->private; // Fatal Error
$obj->printHello(); // Shows Public, Protected and Private


/**
 * Define MyClass2
 */
class MyClass2 extends MyClass
{
    // We can redeclare the public and protected method, but not private
    protected $protected = 'Protected2';

    function printHello()
    {
        echo $this->public;
        echo $this->protected;
        echo $this->private;
    }
}

$obj2 = new MyClass2();
echo $obj->public; // Works
echo $obj2->private; // Undefined
echo $obj2->protected; // Fatal Error
$obj2->printHello(); // Shows Public, Protected2, not Private

?>
```

> **Nota:** La forma de declarar una variable en PHP 4 con la palabra reservada *var* ya no es válida para los objetos de PHP 5. Por compatibilidad un variable declarada en php se asumirá con visibilidad pública, y una se emitirá una advertencia **`E_STRICT`**.

Visibilidad de los métodos

Los métodos de clase deben ser definidos con public, private o protected. Los métodos sin ninguna declaración son tomados como public.

Ejemplo 19-9. Declaración de métodos

```
<?php
/**
 * Define MyClass
 */
class MyClass
{
    // Contructors must be public
    public function __construct() { }

    // Declare a public method
    public function MyPublic() { }

    // Declare a protected method
    protected function MyProtected() { }

    // Declare a private method
    private function MyPrivate() { }

    // This is public
    function Foo()
    {
        $this->MyPublic();
        $this->MyProtected();
        $this->MyPrivate();
    }
}

$myclass = new MyClass;
$myclass->MyPublic(); // Works
$myclass->MyProtected(); // Fatal Error
$myclass->MyPrivate(); // Fatal Error
$myclass->Foo(); // Public, Protected and Private work


/**
 * Define MyClass2
 */
class MyClass2 extends MyClass
{
    // This is public
    function Foo2()
    {
        $this->MyPublic();
        $this->MyProtected();
        $this->MyPrivate(); // Fatal Error
    }
}

$myclass2 = new MyClass2;
$myclass2->MyPublic(); // Works
$myclass2->Foo2(); // Public and Protected work, not Private
?>
```

Alcance del operador de resolución (::)

El alcance del operador de resolución (también llamado Paamayim Nekudotayim) o en términos simples, dobles dos puntos, es un símbolo que permite acceso a los miembros o métodos estaticos,

constantes, y eliminados de una clase.

Cuando se referencín estos elementos desde aufera de la definición de la clase, usan el nombre de la clase.

Paamayin Nekudotayim podrí, en principio, parecer una extraña elección para nombrar un doble-dos-puntos. Sin embargo, mientras se escribia el compilador Zend 0.5, (el cuál da fuerza a PHP 3), ese fue el nombre que el equipo decidió darle. En realidad significa doble-dos-puntos ¡en Hebreo!.

Ejemplo 19-10. :: desde afuera de la definición de la clase

```
<?php
class MyClass {
    const CONST_VALUE = 'A constant value';
}

echo MyClass::CONST_VALUE;
?>
```

Dos palabras reservadas *self* y *parent* son usadas para accesar los miembros o métodos desde adentro de la definición de la clase.

Ejemplo 19-11. :: desde dentro de la definición de la clase

```
<?php
class OtherClass extends MyClass
{
    public static $my_static = 'static var';

    public static function doubleColon() {
        echo parent::CONST_VALUE . "\n";
        echo self::$my_static . "\n";
    }
}

OtherClass::doubleColon();
?>
```

Cuando al extender una clase se elimina las definiciones de un método de la clase padre, PHP no llamará el método de la clase padre. Es opcional a la clase extendida decidir si se deba llamar al método de la clase padre. Esto también aplica a Constructores y Destructores, Sobrecarga, y definición de los métodos Magico.

Ejemplo 19-12. Llamando al método de la clase padre

```
<?php
class MyClass
{
    protected function myFunc() {
        echo "MyClass::myFunc()\n";
    }
}

class OtherClass extends MyClass
{
    // Override parent's definition
    public function myFunc()
    {
        // But still call the parent function
        parent::myFunc();
        echo "OtherClass::myFunc()\n";
    }
}

$class = new OtherClass();
$class->myFunc();
?>
```

La palabra reservada 'Static'

Declarar miembros de clases o métodos como estáticos, los hace accesibles desde afuera del contexto del objeto. Un miembro o método declarado como estático no puede ser accesado con una variable que es una instancia del objeto y no puede ser redefinido en una extensión de la clase.

La declaración static debe estar después de la declaración de visibilidad. Por compatibilidad con PHP 4, si no se usa la declaración de visibilidad, entonces el miembro o método será tratado tal si como se hubiera declarado como *public static*.

A causa de que los métodos estáticos son accesibles sin que se haya creado una instancia del objeto, la pseudo variable *$this* no está disponible dentro de los métodos declarados como estáticos.

De hecho las llamadas a métodos *static* son resueltas en tiempo de ejecución. Cuando se una explícitamente un nombre de clase, el método ya ha sido identificado completamente y no es necesario aplicar las reglas de herencia. Si la llamada es hecha por *self* entonces *self* es traducido a la clase actual, esto es, la clase a la que pertenece el código. Aquí tampoco aplican las reglas de herencia.

Las propiedades estáticas no pueden ser accesadas a través del objeto usando el operador de flecha ->.

Ejemplo 19-13. Ejemplo de miembro Static

```
<?php
class Foo
{
    public static $my_static = 'foo';

    public function staticValue() {
        return self::$my_static;
    }
}

class Bar extends Foo
{
    public function fooStatic() {
        return parent::$my_static;
    }
}


print Foo::$my_static . "\n";

$foo = new Foo();
print $foo->staticValue() . "\n";
print $foo->my_static . "\n";      // Undefined "Property" my_static

// $foo::my_static is not possible

print Bar::$my_static . "\n";
$bar = new Bar();
print $bar->fooStatic() . "\n";
?>
```

Ejemplo 19-14. Ejmeplo de método Static

```php
<?php
class Foo {
    public static function aStaticMethod() {
        // ...
    }
}

Foo::aStaticMethod();
?>
```

Constantes en Objetos

Es posible definir valores constantes en cada clase manteniendo el mismo valor y siendo incambiable. Las constantes difieren de las variables normales en que no se usa el símbolo *$* para declararlas o usarlas. Como los miembros estáticos, los valores constantes no pueden ser accesados desde una instancia de un objeto.

Ejemplo 19-15. Definiendo y usando constantes

```php
<?php
class MyClass
{
    const constant = 'constant value';

    function showConstant() {
        echo  self::constant . "\n";
    }
}

echo MyClass::constant . "\n";

$class = new MyClass();
$class->showConstant();
// echo $class::constant;  is not allowed
?>
```

Abstracción de Objetos

PHP 5 introduce clases y métodos abstractos. No es permitido crear una instancia de una clase que ha sido definida como abstracta. Cualquier clase que contenga por lo menos un método abstracto debe también ser abstracta. Los métodos definidos como abstractos simplemente declaran el método, no pueden definir la implementación

La clase que implementa el método abstracto debe definir con la misma visibilidad o mas débil. Si el método abstracto es definido como protected, la implementació de la función debe ser definida como protected o public.

Ejemplo 19-16. Ejemplos de la clase Abstract

```
<?php
abstract class AbstractClass
{
    // Force Extending class to define this method
    abstract protected function getValue();

    // Common method
    public function printOut() {
        print $this->getValue();
    }
}

class ConcreteClass1 extends AbstractClass
{
    protected function getValue() {
        return "ConcreteClass1";
    }
}

class ConcreteClass2 extends AbstractClass
{
    protected function getValue() {
        return "ConcreteClass2";
    }

}

$class1 = new ConcreteClass1;
$class1->printOut();

$class2 = new ConcreteClass2;
$class2->printOut();
?>
```

El coódigo anterior no tenía clases definidas por el usuario o funciones llamadas 'abstractas' debe correr sin necesidad de modificación.

Interfaces de Objetos

Las interfaces de objetos permiten crear código el cual especifica métodos que una clase debe implementar, sin tener que definir como serán manejados esos métodos.

Las interfaces son definidas usando la palabra reservada 'interface', de la misma manera que las clases estárdar, pero sin que cualquiera de los métodos tenga su contenido definido. Las clases que implementan una interface debeb hacerlo usando la palabra reservada 'implements', y deben tener definiciones para todos los métodos enlistados en la interface. Las clases pueden implementar más de una interface si lo desean, enlistando las interfaces separadas por comas.

Todos los métodos en una interface deben ser publicos, esto es la naturaleza de una interface.

Teniendo que una clase implementa una interface, y no implemente todos los métodos en esa interface, resultará en un error fatal que indica cuales métodos no han sido implementados.

Ejemplo 19-17. Ejemplo de Interface

```php
<?php
// Declare the interface 'iTemplate'
interface iTemplate
{
    public function setVariable($name, $var);
    public function getHtml($template);
}

// Implement the interface
// This will work
class Template implements iTemplate
{
    private $vars = array();

    public function setVariable($name, $var)
    {
        $this->vars[$name] = $var;
    }

    public function getHtml($template)
    {
        foreach($this->vars as $name => $value) {
            $template = str_replace('{' . $name . '}', $value, $template);
        }

        return $template;
    }
}

// This will not work
// Fatal error: Class BadTemplate contains 1 abstract methods
// and must therefore be declared abstract (iTemplate::getHtml)
class BadTemplate implements iTemplate
{
    private $vars = array();

    public function setVariable($name, $var)
    {
        $this->vars[$name] = $var;
    }
}

?>
```

Sobrecarga

Las llamadas a métodos y los accesos a los miembros pueden ser sobrecargadas por medio de los métodos __call, __get y __set. Estos métodos serán accionados cuando su objeto u objeto heredado no contengan los miembros o métodos que está intentado accesar.

Sobrecarga de Miembros

void **__set** (string name, mixed value)

mixed **__get** (mixed name)

Los miembros de la clase pueden ser sobrecargados para ejecutar código personalizado definido en la clase al definir estos métodos de nombre especial. El parámetro *$name* usado es el nombre de la variable que debe ser asignada (set) u obtenida (get). El parámetro *$value* del método __set() especificael calor que el objeto debe tener *$value*.

Ejemplo 19-18. Ejemplo de sobrecarga con with __get y __set

```
<?php
class Setter
{
    public $n;
    private $x = array("a" => 1, "b" => 2, "c" => 3);

    function __get($nm)
    {
        print "Getting [$nm]\n";

        if (isset($this->x[$nm])) {
            $r = $this->x[$nm];
            print "Returning: $r\n";
            return $r;
        } else {
            echo "Nothing!\n";
        }
    }

    function __set($nm, $val)
    {
        print "Setting [$nm] to $val\n";

        if (isset($this->x[$nm])) {
            $this->x[$nm] = $val;
            echo "OK!\n";
        } else {
            echo "Not OK!\n";
        }
    }
}

$foo = new Setter();
$foo->n = 1;
$foo->a = 100;
$foo->a++;
$foo->z++;
var_dump($foo);
?>
```

El resultado del ejemplo seria:

```
Setting [a] to 100
OK!
Getting [a]
Returning: 100
Setting [a] to 101
OK!
Getting [z]
Nothing!
Setting [z] to 1
Not OK!
object(Setter)#1 (2) {
    ["n"]=>
    int(1)
    ["x:private"]=>
    array(3) {
        ["a"]=>
        int(101)
        ["b"]=>
        int(2)
        ["c"]=>
        int(3)
    }
}
```

Sobrecarga de Métodos

mixed **__call** (string name, array arguments)

Los métodos de la clase pueden ser sobrecargados para ejecutar código personalizado definido en la clase al definir este método en particular. El parámetro *$value* es el nombre de la función que se pidió usar. Los argumentos que fueron pasados en la función serán definidos como una matriz en el parámetro *$arguments*. El valor regresado del método __call() será regresado a quien haya llamado al método.

Ejemplo 19-19. Ejemplo de sobrecarga con __call

```
<?php
class Caller
{
    private $x = array(1, 2, 3);

    function __call($m, $a)
    {
        print "Method $m called:\n";
        var_dump($a);
        return $this->x;
    }
}

$foo = new Caller();
$a = $foo->test(1, "2", 3.4, true);
var_dump($a);
?>
```

El resultado del ejemplo seria:

```
Method test called:
array(4) {
    [0]=>
    int(1)
    [1]=>
    string(1) "2"
    [2]=>
    float(3.4)
    [3]=>
    bool(true)
}
array(3) {
    [0]=>
    int(1)
    [1]=>
    int(2)
    [2]=>
    int(3)
}
```

Interacción de Objetos

PHP 5 provee una forma para que los objetos a ser definidos puedan interactuar a través de una lista de campos, con, por ejemplo una sentencia foreach. Por defecto, todas las propiedades visibles serán usadas para la interacción.

Ejemplo 19-20. Simple interacción de un Objeto

```
<?php
class MyClass
{
    public $var1 = 'value 1';
    public $var2 = 'value 2';
    public $var3 = 'value 3';

    protected $protected = 'protected var';
    private   $private   = 'private var';

    function iterateVisible() {
       echo "MyClass::iterateVisible:\n";
       foreach($this as $key => $value) {
           print "$key => $value\n";
       }
    }
}

$class = new MyClass();

foreach($class as $key => $value) {
    print "$key => $value\n";
}
echo "\n";


$class->iterateVisible();

?>
```

El resultado del ejemplo seria:

```
var1 => value 1
var2 => value 2
var3 => value 3

MyClass::iterateVisible:
var1 => value 1
var2 => value 2
var3 => value 3
protected => protected var
private => private var
```

Como se muestra en la salida, foreach interactúa a través de todas la variables visibles que pueden ser accesadas. Para dar un paso más allá, puede *implementar* uno de las interfaces internas de PHP llamada *Iterator*. Esto permite que el objeto decida qué y como el objeto deba interactuar.

Ejemplo 19-21. Interacción de Objetos que implementan Iterator

```php
<?php
class MyIterator implements Iterator
{
    private $var = array();

    public function __construct($array)
    {
        if (is_array($array)) {
            $this->var = $array;
        }
    }

    public function rewind() {
        echo "rewinding\n";
        reset($this->var);
    }

    public function current() {
        $var = current($this->var);
        echo "current: $var\n";
        return $var;
    }

    public function key() {
        $var = key($this->var);
        echo "key: $var\n";
        return $var;
    }

    public function next() {
        $var = next($this->var);
        echo "next: $var\n";
        return $var;
    }

    public function valid() {
        $var = $this->current() !== false;
        echo "valid: {$var}\n";
        return $var;
    }
}

$values = array(1,2,3);
$it = new MyIterator($values);

foreach ($it as $a => $b) {
    print "$a: $b\n";
}
?>
```

El resultado del ejemplo seria:

```
rewinding
current: 1
valid: 1
current: 1
key: 0
0: 1
next: 2
current: 2
valid: 1
current: 2
key: 1
1: 2
next: 3
current: 3
valid: 1
current: 3
key: 2
2: 3
next:
current:
valid:
```

También puede definir la clase de tal manera que no necesita definir todas la funciones del *Iterator*, simplemente implementando la interface PHP 5 *IteratorAggregate*.

Ejemplo 19-22. Interacción de objetos que implementan IteratorAggregate

```
<?php
class MyCollection implements IteratorAggregate
{
    private $items = array();
    private $count = 0;

    // Required definition of interface IteratorAggregate
    public function getIterator() {
        return new MyIterator($this->items);
    }

    public function add($value) {
        $this->items[$this->count++] = $value;
    }
}

$coll = new MyCollection();
$coll->add('value 1');
$coll->add('value 2');
$coll->add('value 3');

foreach ($coll as $key => $val) {
    echo "key/value: [$key -> $val]\n\n";
}
?>
```

El resultado del ejemplo seria:

```
rewinding
current: value 1
valid: 1
current: value 1
key: 0
key/value: [0 -> value 1]

next: value 2
current: value 2
valid: 1
current: value 2
key: 1
key/value: [1 -> value 2]

next: value 3
current: value 3
valid: 1
current: value 3
key: 2
key/value: [2 -> value 3]

next:
current:
valid:
```

Nota: Para más ejemplos de iterators, vea la Extensión SPL.

Patrones

Los patrones son formas de describir las mejores prácticas y los buenos diseños. Estos muestran una solución flexible a los problemas comunes de programación.

Factory

El patrón Factory permita la instancia de objetos en tiempo de ejecución. Es llamado el patr6oacute;n Factory puesto que es responsable de "manufacturar" un objeto.

Ejemplo 19-23. Método Factory

```
<?php
class Example
{
    // The factory method
    public static function &factory($type)
    {
        if (include_once 'Drivers/' . $type . '.php') {
            $classname = 'Driver_' . $type;
            return new $classname;
        } else {
            throw new Exception ('Driver not found');
        }
    }
}
?>
```

Al definir este método en una clase se nos permite que los drivers sean cargados al vuelo. Si la clase *Example* fuera una clase de abstracción de base de datos, cargar un manejador de *MySQL* y *SQLite* podría ser hecho como sigue:

```php
<?php
// Load a MySQL Driver
$mysql = Example::factory('MySQL');

// Load a SQLite Driver
$sqlite = Example::factory('SQLite');
?>
```

Singleton

El patrón Singleton aplica a situaciones en las cuales hay la necesidad de ser una sola instancia de una clase. El ejemplo más común de esto es una conexión de base de datos. Implementando este patrón permite a un programador hacer esta simple isntancia fácilmente accesible a muchos otros objetos.

Ejemplo 19-24. Función Singleton

```php
<?php
class Example
{
    // Hold an instance of the class
    private static $instance;

    // A private constructor; prevents direct creation of object
    private function __construct()
    {
        echo 'I am constructed';
    }

    // The singleton method
    public static function singleton()
    {
        if (!isset(self::$instance)) {
            $c = __CLASS__;
            self::$instance = new $c;
        }

        return self::$instance;
    }

    // Example method
    public function bark()
    {
        echo 'Woof!';
    }

    // Prevent users to clone the instance
    public function __clone()
    {
        trigger_error('Clone is not allowed.', E_USER_ERROR);
    }

}

?>
```

Esto permite que se obtenga una simple instancia de la clase *Example*.

```php
<?php
// This would fail because the constructor is private
$test = new Example;

// This will always retrieve a single instance of the class
$test = Example::singleton();
$test->bark();

// This will issue an E_USER_ERROR.
$test_clone = clone($test);

?>
```

Métodos mágicos

Los nombres de función: *__construct*, *__destruct* (see Constructores y Destructores), *__call*, *__get*, *__set* (see Sobrecarga), *__sleep*, *__wakeup*, y *__toString* son mágicos en las clases de PHP. No puede tener funciones con esos nombres en cualquiera de sus clases a menos que se desee la funcionalidad mágica asociada con ellos.

Atención
PHP reserva todos los nombres de funciones que empiecen con __ como mágicas. Es recomendado que no use nombres de funciones con __ en PHP a menos que dese alguna mágica funcionalidad documentada.

__sleep y *__wakeup*

serialize() checa si su clase tiene una función con el nombre má *__sleep*. Si es así, esa función es ejecutada antes de cualquier serialización. Esta puede limpiar el objeto y se espera que regrese una matriz con los nombres de todas las variables de ese objeto que puede ser serializadas.

La intención de usar *__sleep* es cerrar cualquier conexión a base de datos que el objeto pueda tener, terminar de enviar cualquier dato o ejecutar tareas similares de limpieza. También, la función es útil si tiene objetos muy grandes los cuales no necesitan mantenerse completos.

Inversamente, **unserialize()** checa por la presencia de una función con el nombre mágico *__wakeup*. Si está presente, está función puede reconstruir cualquier recurso que el objeto pueda tener.

La intención de *__wakeup* es reestablecer cualquier conexión a base de datos que se pueda haber perdido durante la serialización y ejecutar otras tareas de reinicialización.

__toString

El método *__toString* permite a una clase decidir como actuar cuando es convertida en cadena.

Ejemplo 19-25. Ejemplo simple

```php
<?php
// Declare a simple class
class TestClass
{
    public $foo;

    public function __construct($foo) {
        $this->foo = $foo;
    }

    public function __toString() {
        return $this->foo;
    }
}

$class = new TestClass('Hello');
echo $class;
?>
```

El resultado del ejemplo seria:

```
Hello
```

No tiene otro valor que cuando el método *__toString* es llamado solo cuando es directamente combinado con **echo()** o **print()**.

Ejemplo 19-26. Casos donde *__toString* es llamado

```php
<?php
// __toString called
echo $class;

// __toString called (still a normal parameter for echo)
echo 'text', $class;

// __toString not called (concatenation operator used first)
echo 'text' . $class;

// __toString not called (cast to string first)
echo (string) $class;

// __toString not called (cast to string first)
echo "text $class";
?>
```

La palabra reservada 'Final'

PHP 5 introduce la palabra reservada 'final', la cual prevee que las clases hijo puedan sobreescribir un método, usando el prefijo 'final' en la definición del método. Si la clase en sí misma es definida como 'final' entonces no puede ser extendida.

Ejemplo 19-27. Ejemplo de métodos con Final

```
<?php
class BaseClass {
   public function test() {
       echo "BaseClass::test() called\n";
   }

   final public function moreTesting() {
       echo "BaseClass::moreTesting() called\n";
   }
}

class ChildClass extends BaseClass {
   public function moreTesting() {
       echo "ChildClass::moreTesting() called\n";
   }
}
// Results in Fatal error: Cannot override final method BaseClass::moreTesting()
?>
```

Ejemplo 19-28. Ejemplo de clase con Final

```
<?php
final class BaseClass {
   public function test() {
       echo "BaseClass::test() called\n";
   }

   // Here it doesn't matter if you specify the function as final or not
   final public function moreTesting() {
       echo "BaseClass::moreTesting() called\n";
   }
}

class ChildClass extends BaseClass {
}
// Results in Fatal error: Class ChildClass may not inherit from final class (BaseClass
?>
```

Clonado de Objetos

Crear una copia de un objeto con una replica de todas sus propiedades no es siempre lo que se desea hacer. Un buen ejemplo de la necesidad de copiar los constructores, es si se tiene un objeto el cual representa una ventana GTK y el objeto contiene los recursos de esta ventana GTK, cuando se crea un duplicado, puede querese crear una nueva ventana con las mismas propiedades y hacer que el nuevo objeto tenga los recursos de la ventana nueva. Otro ejemplo es si su objeto tiene la referencia a otro objeto el cual usa, y cuando se duplica el objeto padre, quiere crear una nueva instancia de este otro objeto así que la replica tiene su propia copia.

Una copia de un objeto es creada usando la palabra 'clone' (la cual llama el método __clone() del objeto, si es posible). Un método __clone() de un objeto no puede ser llamado directamente.

```
$copy_of_object = clone $object;
```

Cuando un objeto es clonado, PHP 5 informará una copia baja de todas las propiedades del objeto. Cualquier propiedad que sean referencias a otras variables, permanecerán siendo referencias. Si un método __clone() es definido, entonces el método __clone() del nuevo objeto creado será llamado, para permitir cualquier propiedad que tenga que ser cambiada.

Ejemplo 19-29. Clonando un objeto

```php
<?php
class SubObject
{
    static $instances = 0;
    public $instance;

    public function __construct() {
        $this->instance = ++self::$instances;
    }

    public function __clone() {
        $this->instance = ++self::$instances;
    }
}

class MyCloneable
{
    public $object1;
    public $object2;

    function __clone()
    {
        // Force a copy of this->object, otherwise
        // it will point to same object.
        $this->object1 = clone($this->object1);
    }
}

$obj = new MyCloneable();

$obj->object1 = new SubObject();
$obj->object2 = new SubObject();

$obj2 = clone $obj;


print("Original Object:\n");
print_r($obj);

print("Cloned Object:\n");
print_r($obj2);

?>
```

El resultado del ejemplo seria:

```
Original Object:
MyCloneable Object
(
    [object1] => SubObject Object
        (
            [instance] => 1
        )

    [object2] => SubObject Object
        (
            [instance] => 2
        )

)
Cloned Object:
MyCloneable Object
(
    [object1] => SubObject Object
        (
            [instance] => 3
        )

    [object2] => SubObject Object
        (
            [instance] => 2
        )

)
```

Comparación de Objetos

En PHP 5, la comparación de objetos es más complicada que en PHP 4 y más de acuerdo con lo que uno esperaría de un lenguaje orientado a objetos (no es que PHP no sea uno de tales lenguajes).

Cuando se usa el operador de comparación (==), las variables del objeto son comparadas de una forma simple, digase: Dos instancias de objetos son iguales si tienes los mismos atributos y valores, y son instancias de la misma clase.

Por otro lado, cuando se usa el operador identico (===), las variables del objeto son identicas solo si refieren a la misma instancia de la misma clase.

Un ejemplo clarificará estas reglas.
Ejemplo 19-30. Ejemplo de comparación de objetos en PHP 5

```php
<?php
function bool2str($bool)
{
    if ($bool === false) {
        return 'FALSE';
    } else {
        return 'TRUE';
    }
}

function compareObjects(&$o1, &$o2)
{
    echo 'o1 == o2 : ' . bool2str($o1 == $o2) . "\n";
    echo 'o1 != o2 : ' . bool2str($o1 != $o2) . "\n";
    echo 'o1 === o2 : ' . bool2str($o1 === $o2) . "\n";
    echo 'o1 !== o2 : ' . bool2str($o1 !== $o2) . "\n";
}

class Flag
{
    public $flag;

    function Flag($flag = true) {
        $this->flag = $flag;
    }
}

class OtherFlag
{
    public $flag;

    function OtherFlag($flag = true) {
        $this->flag = $flag;
    }
}

$o = new Flag();
$p = new Flag();
$q = $o;
$r = new OtherFlag();

echo "Two instances of the same class\n";
compareObjects($o, $p);

echo "\nTwo references to the same instance\n";
compareObjects($o, $q);

echo "\nInstances of two different classes\n";
compareObjects($o, $r);
?>
```

El resultado del ejemplo seria:

```
Two instances of the same class
o1 == o2 : TRUE
o1 != o2 : FALSE
o1 === o2 : FALSE
o1 !== o2 : TRUE

Two references to the same instance
o1 == o2 : TRUE
o1 != o2 : FALSE
o1 === o2 : TRUE
o1 !== o2 : FALSE

Instances of two different classes
o1 == o2 : FALSE
o1 != o2 : TRUE
o1 === o2 : FALSE
o1 !== o2 : TRUE
```

Reflección

Introduction

PHP 5 viene con un API completa de reflexión que agrega la habilidad de hacer ingeniería inversa de clases, interfaces, funciones y métodos así como extensiones. Adicionalmente, el API de reflexión también ofrece formas de obtener los comentarios de los documentos para funciones, clases y métodos.

El API de reflexión es una extensión orientada a objetos para el compilador Zend, consistente de las siguientes clases:

```
<?php
class Reflection { }
interface Reflector { }
class ReflectionException extends Exception { }
class ReflectionFunction implements Reflector { }
class ReflectionParameter implements Reflector { }
class ReflectionMethod extends ReflectionFunction { }
class ReflectionClass implements Reflector { }
class ReflectionObject extends ReflectionClass { }
class ReflectionProperty implements Reflector { }
class ReflectionExtension implements Reflector { }
?>
```

Nota: Para detalles de estas clases, de una mirada a los siguientes capítulos.

Si fuéramos a ejecutar el código en el siguiente ejemplo:

Ejemplo 19-31. Uso básico del API reflexión

```
<?php
Reflection::export(new ReflectionClass('Exception'));
?>
```

El resultado del ejemplo seria:

```
Class [ <internal> class Exception ] {

  - Constants [0] {
  }

  - Static properties [0] {
  }

  - Static methods [0] {
  }

  - Properties [6] {
    Property [ <default> protected $message ]
    Property [ <default> private $string ]
    Property [ <default> protected $code ]
    Property [ <default> protected $file ]
    Property [ <default> protected $line ]
    Property [ <default> private $trace ]
  }

  - Methods [9] {
    Method [ <internal> final private method __clone ] {
    }

    Method [ <internal> <ctor> public method __construct ] {

      - Parameters [2] {
        Parameter #0 [ <required> $message ]
        Parameter #1 [ <required> $code ]
      }
    }

    Method [ <internal> final public method getMessage ] {
    }

    Method [ <internal> final public method getCode ] {
    }

    Method [ <internal> final public method getFile ] {
    }

    Method [ <internal> final public method getLine ] {
    }

    Method [ <internal> final public method getTrace ] {
    }

    Method [ <internal> final public method getTraceAsString ] {
    }

    Method [ <internal> public method __toString ] {
    }
  }
}
```

ReflectionFunction

La clase **ReflectionFunction** te permite funciones de ingenierí inversa.

```
<?php
class ReflectionFunction implements Reflector
{
    final private __clone()
    public object __construct(string name)
    public string __toString()
    public static string export()
    public string getName()
    public bool isInternal()
    public bool isUserDefined()
    public string getFileName()
    public int getStartLine()
    public int getEndLine()
    public string getDocComment()
    public array getStaticVariables()
    public mixed invoke(mixed* args)
    public mixed invokeArgs(array args)
    public bool returnsReference()
    public ReflectionParameter[] getParameters()
    public int getNumberOfParameters()
    public int getNumberOfRequiredParameters()
}
?>
```

Nota: invokeArgs() fue agregado en PHP 5.1.0.

Para entender directamente una función, primero tiene que crear una isntancia de la clase **ReflectionFunction**. Hasta entonces puede llamar cualquier de los métodos anteriores en esta instancia.

Ejemplo 19-32. Usando la clase ReflectionFunction

```php
<?php
/**
 * A simple counter
 *
 * @return    int
 */
function counter()
{
    static $c = 0;
    return $c++;
}

// Create an instance of the Reflection_Function class
$func = new ReflectionFunction('counter');

// Print out basic information
printf(
    "===> The %s function '%s'\n".
    "     declared in %s\n".
    "     lines %d to %d\n",
    $func->isInternal() ? 'internal' : 'user-defined',
    $func->getName(),
    $func->getFileName(),
    $func->getStartLine(),
    $func->getEndline()
);

// Print documentation comment
printf("---> Documentation:\n %s\n", var_export($func->getDocComment(), 1));

// Print static variables if existant
if ($statics = $func->getStaticVariables())
{
    printf("---> Static variables: %s\n", var_export($statics, 1));
}

// Invoke the function
printf("---> Invokation results in: ");
var_dump($func->invoke());


// you may prefer to use the export() method
echo "\nReflectionFunction::export() results:\n";
echo ReflectionFunction::export('counter');
?>
```

Nota: El método **invoke()** acepta un número de variable de argumentos los cuales son pasados a la función tal y como se hace en **call_user_func()**.

ReflectionParameter

La case **ReflectionParameter** obtiene información acerca de los parámetros de una función o un método.

```php
<?php
class ReflectionParameter implements Reflector
{
    final private __clone()
    public object __construct(string name)
    public string __toString()
    public static string export()
    public string getName()
    public bool isPassedByReference()
    public ReflectionClass getClass()
    public bool allowsNull()
    public bool isOptional()
    public bool isDefaultValueAvailable()
    public mixed getDefaultValue()
}
?>
```

Nota: getDefaultValue(), **isDefaultValueAvailable()**, **isOptional()** fueron agregados en PHP 5.1.0.

Para entender los parámetros de la función, tendrá primero que crear una instancia de la clase **ReflectionFunction** o de la clase **ReflectionMethod** y entonces usar sus método **getParameters()** para obtener una matriz de parámetros.

Ejemplo 19-33. Usando la clase ReflectionParameter

```php
<?php
function foo($a, $b, $c) { }
function bar(Exception $a, &$b, $c) { }
function baz(ReflectionFunction $a, $b = 1, $c = null) { }
function abc() { }

// Create an instance of Reflection_Function with the
// parameter given from the command line.
$reflect = new ReflectionFunction($argv[1]);

echo $reflect;

foreach ($reflect->getParameters() as $i => $param) {
    printf(
        "-- Parameter #%d: %s {\n".
        "   Class: %s\n".
        "   Allows NULL: %s\n".
        "   Passed to by reference: %s\n".
        "   Is optional?: %s\n".
        "}\n",
        $i,
        $param->getName(),
        var_export($param->getClass(), 1),
        var_export($param->allowsNull(), 1),
        var_export($param->isPassedByReference(), 1),
        $param->isOptional() ? 'yes' : 'no'
    );
}
?>
```

ReflectionClass

La clase **ReflectionClass** te permite hacer ingeniería inversa de clases.

```
<?php
class ReflectionClass implements Reflector
{
    final private __clone()
    public object __construct(string name)
    public string __toString()
    public static string export()
    public string getName()
    public bool isInternal()
    public bool isUserDefined()
    public bool isInstantiable()
    public bool hasMethod(string name)
    public string getFileName()
    public int getStartLine()
    public int getEndLine()
    public string getDocComment()
    public ReflectionMethod getConstructor()
    public ReflectionMethod getMethod(string name)
    public ReflectionMethod[] getMethods()
    public ReflectionProperty getProperty(string name)
    public ReflectionProperty[] getProperties()
    public array getConstants()
    public mixed getConstant(string name)
    public ReflectionClass[] getInterfaces()
    public bool isInterface()
    public bool isAbstract()
    public bool isFinal()
    public int getModifiers()
    public bool isInstance(stdclass object)
    public stdclass newInstance(mixed* args)
    public ReflectionClass getParentClass()
    public bool isSubclassOf(ReflectionClass class)
    public array getStaticProperties()
    public array getDefaultProperties()
    public bool isIterateable()
    public bool implementsInterface(string name)
    public ReflectionExtension getExtension()
    public string getExtensionName()
}
?>
```

Nota: hasMethod() fue agregado en PHP 5.1.0.

Para entender una clase, primero tendrá que crear una instancia de la clase **ReflectionClass**. Entonces puede llamar cualquier de los métodos anteriores en esta instancia.

Ejemplo 19-34. Usando la clase ReflectionClass

```php
<?php
interface Serializable
{
    // ...
}

class Object
{
    // ...
}

/**
 * A counter class
 */
class Counter extends Object implements Serializable
{
    const START = 0;
    private static $c = Counter::START;

    /**
     * Invoke counter
     *
     * @access  public
     * @return  int
     */
    public function count() {
        return self::$c++;
    }
}

// Create an instance of the ReflectionClass class
$class = new ReflectionClass('Counter');

// Print out basic information
printf(
    "===> The %s%s%s %s '%s' [extends %s]\n" .
    "     declared in %s\n" .
    "     lines %d to %d\n" .
    "     having the modifiers %d [%s]\n",
        $class->isInternal() ? 'internal' : 'user-defined',
        $class->isAbstract() ? ' abstract' : '',
        $class->isFinal() ? ' final' : '',
        $class->isInterface() ? 'interface' : 'class',
        $class->getName(),
        var_export($class->getParentClass(), 1),
        $class->getFileName(),
        $class->getStartLine(),
        $class->getEndline(),
        $class->getModifiers(),
        implode(' ', Reflection::getModifierNames($class->getModifiers()))
);

// Print documentation comment
printf("---> Documentation:\n %s\n", var_export($class->getDocComment(), 1));

// Print which interfaces are implemented by this class
printf("---> Implements:\n %s\n", var_export($class->getInterfaces(), 1));

// Print class constants
printf("---> Constants: %s\n", var_export($class->getConstants(), 1));

// Print class properties
printf("---> Properties: %s\n", var_export($class->getProperties(), 1));

// Print class methods
printf("---> Methods: %s\n", var_export($class->getMethods(), 1));

// If this class is instantiable, create an instance
if ($class->isInstantiable()) {
    $counter = $class->newInstance();

    echo '---> $counter is instance? ';
```

Nota: El método **newInstance()** acepta un número variable de argumentos los cuales son pasados a la función tal y como si se usara **call_user_func()**.

Nota: *$class = new ReflectionClass('Foo'); $class->isInstance($arg)* es ñalente a *$arg instanceof Foo* o *is_a($arg, 'Foo')*.

ReflectionMethod

La clase **ReflectionMethod** te permite hacer ingenieria inversa de los métodos de la clase.

```
<?php
class ReflectionMethod extends ReflectionFunction
{
    public __construct(mixed class, string name)
    public string __toString()
    public static string export()
    public mixed invoke(stdclass object, mixed* args)
    public moxed invokeArgs(stdclass object, array args)
    public bool isFinal()
    public bool isAbstract()
    public bool isPublic()
    public bool isPrivate()
    public bool isProtected()
    public bool isStatic()
    public bool isConstructor()
    public bool isDestructor()
    public int getModifiers()
    public ReflectionClass getDeclaringClass()

    // Inherited from ReflectionFunction
    final private __clone()
    public string getName()
    public bool isInternal()
    public bool isUserDefined()
    public string getFileName()
    public int getStartLine()
    public int getEndLine()
    public string getDocComment()
    public array getStaticVariables()
    public bool returnsReference()
    public ReflectionParameter[] getParameters()
    public int getNumberOfParameters()
    public int getNumberOfRequiredParameters()
}
?>
```

Para entender los métodos, primero tendrá que crear una instancia de la clase **ReflectionMethod**. Puede entonces llamar cualquiera de los métodos anteriores en esta instancia.

Ejemplo 19-35. Usando la clase ReflectionMethod

```php
<?php
class Counter
{
    private static $c = 0;

    /**
     * Increment counter
     *
     * @final
     * @static
     * @access  public
     * @return  int
     */
    final public static function increment()
    {
        return ++self::$c;
    }
}

// Create an instance of the Reflection_Method class
$method = new ReflectionMethod('Counter', 'increment');

// Print out basic information
printf(
    "===> The %s%s%s%s%s%s%s method '%s' (which is %s)\n" .
    "     declared in %s\n" .
    "     lines %d to %d\n" .
    "     having the modifiers %d[%s]\n",
        $method->isInternal() ? 'internal' : 'user-defined',
        $method->isAbstract() ? ' abstract' : '',
        $method->isFinal() ? ' final' : '',
        $method->isPublic() ? ' public' : '',
        $method->isPrivate() ? ' private' : '',
        $method->isProtected() ? ' protected' : '',
        $method->isStatic() ? ' static' : '',
        $method->getName(),
        $method->isConstructor() ? 'the constructor' : 'a regular method',
        $method->getFileName(),
        $method->getStartLine(),
        $method->getEndline(),
        $method->getModifiers(),
        implode(' ', Reflection::getModifierNames($method->getModifiers()))
);

// Print documentation comment
printf("---> Documentation:\n %s\n", var_export($method->getDocComment(), 1));

// Print static variables if existant
if ($statics= $method->getStaticVariables()) {
    printf("---> Static variables: %s\n", var_export($statics, 1));
}

// Invoke the method
printf("---> Invokation results in: ");
var_dump($method->invoke(NULL));
?>
```

Nota: Tratar de invocar métodos private, protected o abstract resultará en una excepción siendo arrojada del método **invoke()**.

Nota: Para métodos static como se vió anteriormente, se debe a **invoke()** se debe pasar NULL como primer argumento. Para métodos no estáticos, se pasa una instancia de la clase.

ReflectionProperty

La clase **ReflectionProperty** te permite hacer ingeniería inversa a las propiedades de la clase.

```
<?php
class ReflectionProperty implements Reflector
{
    final private __clone()
    public __construct(mixed class, string name)
    public string __toString()
    public static string export()
    public string getName()
    public bool isPublic()
    public bool isPrivate()
    public bool isProtected()
    public bool isStatic()
    public bool isDefault()
    public int getModifiers()
    public mixed getValue(stdclass object)
    public void setValue(stdclass object, mixed value)
    public ReflectionClass getDeclaringClass()
}
?>
```

Para entender las propiedades, se debe primero crear una instancia de la clase **ReflectionProperty**. Y entonces puede llamar cualquiera de los métodos anteriores sobre esta instancia.

Ejemplo 19-36. Usando la clase ReflectionProperty

```
<?php
class String
{
    public $length  = 5;
}

// Create an instance of the ReflectionProperty class
$prop = new ReflectionProperty('String', 'length');

// Print out basic information
printf(
    "===> The%s%s%s%s property '%s' (which was %s)\n" .
    "     having the modifiers %s\n",
        $prop->isPublic() ? ' public' : '',
        $prop->isPrivate() ? ' private' : '',
        $prop->isProtected() ? ' protected' : '',
        $prop->isStatic() ? ' static' : '',
        $prop->getName(),
        $prop->isDefault() ? 'declared at compile-time' : 'created at run-time',
        var_export(Reflection::getModifierNames($prop->getModifiers()), 1)
);

// Create an instance of String
$obj= new String();

// Get current value
printf("---> Value is: ");
var_dump($prop->getValue($obj));

// Change value
$prop->setValue($obj, 10);
printf("---> Setting value to 10, new value is: ");
var_dump($prop->getValue($obj));

// Dump object
var_dump($obj);
?>
```

> **Nota:** Trying to get or set private or protected class property's values will result in an exception being thrown.

ReflectionExtension

La clase **ReflectionExtension** te permite hacer ingeniería inversa a extensiones. Puede obtener todas las extensiones cargadas en tiempo de ejecución usando **get_loaded_extensions()**.

```
<?php
class ReflectionExtension implements Reflector {
    final private __clone()
    public __construct(string name)
    public string __toString()
    public static string export()
    public string getName()
    public string getVersion()
    public ReflectionFunction[] getFunctions()
    public array getConstants()
    public array getINIEntries()
    public ReflectionClass[] getClasses()
    public array getClassNames()
}
?>
```

Para entender una extensión, primero se tiene que crear una instancia de la clase **ReflectionExtension**. Y entonces puede llamarse a cualquiera de los étodos mencionados arriba sobre esa instancia.

Ejemplo 19-37. Usando la clase ReflectionExtension

```
<?php
// Create an instance of the ReflectionProperty class
$ext = new ReflectionExtension('standard');

// Print out basic information
printf(
    "Name        : %s\n" .
    "Version     : %s\n" .
    "Functions   : [%d] %s\n" .
    "Constants   : [%d] %s\n" .
    "INI entries : [%d] %s\n" .
    "Classes     : [%d] %s\n",
        $ext->getName(),
        $ext->getVersion() ? $ext->getVersion() : 'NO_VERSION',
        sizeof($ext->getFunctions()),
        var_export($ext->getFunctions(), 1),

        sizeof($ext->getConstants()),
        var_export($ext->getConstants(), 1),

        sizeof($ext->getINIEntries()),
        var_export($ext->getINIEntries(), 1),

        sizeof($ext->getClassNames()),
        var_export($ext->getClassNames(), 1)
);
?>
```

Extendiendo las clases de reflexión

EN caso de que se quiera crear una versión especializada de las clases integradas (es decir, para crear HTML con colores cuando se exporta, tener fácil acceso a las variables de los miembros en lugar de los métodos o tener métodos de utilería), se puede simplemente extenderlos.

Ejemplo 19-38. Extendiendo las clase integradas

```php
<?php
/**
 * My Reflection_Method class
 */
class My_Reflection_Method extends ReflectionMethod
{
    public $visibility = '';

    public function __construct($o, $m)
    {
        parent::__construct($o, $m);
        $this->visibility= Reflection::getModifierNames($this->getModifiers());
    }
}

/**
 * Demo class #1
 *
 */
class T {
    protected function x() {}
}

/**
 * Demo class #2
 *
 */
class U extends T {
    function x() {}
}

// Print out information
var_dump(new My_Reflection_Method('U', 'x'));
?>
```

Nota: Precaución: Si se desea sobreescribir el constructor, recuerde llamar el constructor padre _antes_ que cualquier otro código que se inserte. El no hacerlo así resultará en: *Fatal error: Internal error: Failed to retrieve the reflection object*

Type Hinting

PHP 5 introduce Type Hinting. Las funciones ahora son capaces de forzar que los parámetros sean objetos especificando el nombre de la clase en el prototipo de la función.

Ejemplo 19-39. Ejemplo de Type Hinting

```php
<?php
// An example class
class MyClass
{
    /**
     * A test function
     *
     * First parameter must be an object of type OtherClass
     */
    public function test(OtherClass $otherclass) {
        echo $otherclass->var;
    }
}

// Another example class
class OtherClass {
    public $var = 'Hello World';
}
?>
```

Al no satisfacer el tipo al que se le hace referencia resulta en un error fatal.

```
<?php
// An instance of each class
$myclass = new MyClass;
$otherclass = new OtherClass;

// Fatal Error: Argument 1 must be an object of class OtherClass
$myclass->test('hello');

// Fatal Error: Argument 1 must be an instance of OtherClass
$foo = new stdClass;
$myclass->test($foo);

// Fatal Error: Argument 1 must not be null
$myclass->test(null);

// Works: Prints Hello World
$myclass->test($otherclass);
?>
```

Type hinting también aplica en funciones:

```
<?php
// An example class
class MyClass {
    public $var = 'Hello World';
}

/**
 * A test function
 *
 * First parameter must be an object of type MyClass
 */
function MyFunction (MyClass $foo) {
    echo $foo->var;
}

// Works
$myclass = new MyClass;
MyFunction($myclass);
?>
```

Type Hints puede solo ser del tipo **object**. El tradicional type hinting con **int** y **string** no está permitidos.

Capítulo 20. Excepciones

PHP 5 tiene un modelo de excepciones similar al de otros lenguajes de programación. Una excepción puede ser lanzada, intentada o capturada en PHP. Un bloque de intento (try) debe incluir por lo menos un bloque de captura (catch). Los bloques de captura múltiples pueden ser usados para capturar diferentes tipos de clases; la ejecución continuará después del último bloque de captura definido. Las excepciones pueden ser lanzadas dentro de bloques de captura.

Cuando es lanzada una excepción, la siguiente línea de código no será ejecutada y PHP intentará encontrar el primer bloque de captura de excepciones. Si una excepción no es capturada se despliega un error fatal de PHP con un mensaje de que la excepción no fue capturada, a menos que exista un manejador de errores definido como **set_exception_handler()**.

Ejemplo 20-1. Lanzando una Excepción

```php
<?php
try {
    $error = 'Always throw this error';
    throw new Exception($error);

    // Code following an exception is not executed.
    echo 'Never executed';

} catch (Exception $e) {
    echo 'Caught exception: ',  $e->getMessage(), "\n";
}

// Continue execution
echo 'Hello World';
?>
```

Extendiendo excepciones

Una clase de definición de excepciones del usuario puede ser definida extendiendo la clase de excepciones incorporado. Los miembros y propiedades mencionados en seguida, muestran lo que está accesible dentro de la clase "hijo" que se deriba de la clase de excepciones incorporados.

Ejemplo 20-2. La clase de excepciones incorporada

```php
<?php
class Exception
{
    protected $message = 'Unknown exception';   // exception message
    protected $code = 0;                        // user defined exception code
    protected $file;                            // source filename of exception
    protected $line;                            // source line of exception

    function __construct($message = null, $code = 0);

    final function getMessage();                // message of exception
    final function getCode();                   // code of exception
    final function getFile();                   // source filename
    final function getLine();                   // source line
    final function getTrace();                  // an array of the backtrace()
    final function getTraceAsString();          // formated string of trace

    /* Overrideable */
    function __toString();                       // formated string for display
}
?>
```

Si una clase se extiende de la clase Exception incorporada y redefine el constructor, es altamente recomendado que también llame parent::__construct() para asegurarse que todos los datos disponibles han sido asignados apropiadamente. El método __toString() puede ser evitado para proveer una salida personalizada cuando el objeto es presentado como una cadena.

Ejemplo 20-3. Extendiendo la clase Exception

```php
<?php
/**
 * Define a custom exception class
 */
class MyException extends Exception
{
    // Redefine the exception so message isn't optional
    public function __construct($message, $code = 0) {
        // some code

        // make sure everything is assigned properly
        parent::__construct($message, $code);
    }

    // custom string representation of object */
    public function __toString() {
        return __CLASS__ . ": [{$this->code}]: {$this->message}\n";
    }

    public function customFunction() {
        echo "A Custom function for this type of exception\n";
    }
}


/**
 * Create a class to test the exception
 */
class TestException
{
    public $var;

    const THROW_NONE    = 0;
    const THROW_CUSTOM  = 1;
    const THROW_DEFAULT = 2;

    function __construct($avalue = self::THROW_NONE) {

        switch ($avalue) {
            case self::THROW_CUSTOM:
                // throw custom exception
                throw new MyException('1 is an invalid parameter', 5);
                break;

            case self::THROW_DEFAULT:
                // throw default one.
                throw new Exception('2 isnt allowed as a parameter', 6);
                break;

            default:
                // No exception, object will be created.
                $this->var = $avalue;
                break;
        }
    }
}


// Example 1
try {
    $o = new TestException(TestException::THROW_CUSTOM);
} catch (MyException $e) {      // Will be caught
    echo "Caught my exception\n", $e;
    $e->customFunction();
} catch (Exception $e) {        // Skipped
    echo "Caught Default Exception\n", $e;
}

// Continue execution
var_dump($o);
echo "\n\n";
```

Capítulo 21. Explicando las Referencias

What References Are

Las Referencias en PHP son un medio para acceder al mismo contenido de una variable pero con diferentes nombres. No son como los punteros de C, sino que son alias en la tabla de símbolos. Hay que tener en cuenta, que en PHP el nombre de una variable y el contenido de una variable son diferentes, de manera que el mismo contenido, puede tener varios nombres. La analogía más cercana podría ser la de los archivos de Unix y sus nombres, dónde los nombres de las variables serían los directorios, y el contenido de las variables es el archivo en si. Las referencias también pueden ser pensadas como un enlace duro en un sistema de archivos Unix.

Lo que las Referencias son

Las Referencias en PHP te permiten lograr que dos variables "apunten" al mismo contenido. Cuando haces algo como:

```
$a =& $b
```

significa que *$a* y *$b* apuntan a la misma variable.

> **Nota:** *$a* y *$b* son completamente iguales, no es que *$a* esté apuntando a *$b* o viceversa, sino que tanto *$a* como *$b* apuntan al mismo lugar.

La misma sintáxis puede ser utilizada con funciones, que devuelven Referencias, y con el operador *new* (en PHP 4.0.4 o superior):

```
$bar =& new fooclass();
$foo =& find_var ($bar);
```

> **Nota:** El no utilizar el operador *&* causa que el objeto sea copiado en memoria. Si utilizamos *$this* en la clase, entonces actuaremos sobre la instancia actual de la clase. Las asignaciones sin *&* harán una copia de la instancia (por ejemplo, del objeto) y *$this* operará en la copia, lo que no siempre es el comportamiento deseado. Usualmente se desea utilizar una sola instancia, debido a razones de memoria y performance de la aplicación.
>
> Mientras que se puede utilizar @ para *silenciar* cualquier error en el constructor utilizando *@new*, esto no funciona cuando utilizamos *&new*. Esto es una limitación del Zend Engine y por lo tanto, resultará en un error de sintáxis.

Otro uso que se le puede dar a las referencias es el traspaso de variables por-referencia. Esto se logra haciendo que una variable 'local' a la función y una variable en el script 'referencien' al mismo contenido. Por ejemplo:

```
function foo (&$var)
{
    $var++;
}

$a=5;
foo ($a);
```

hará que *$a* valga 6. Esto es posible porque en la funció *foo*, la variable *$var* 'referencia' al mismo contenido que la variable *$a*. Más información acerca de paso por referencia.

Un tercer uso de las referencias es el retorno por referencia.

Lo que las Referencias no son

Como se ha mencionado antes, las Referencias NO son punteros. Esto significa que el siguiente ejemplo no hará lo que se espera:

```
function foo (&$var)
{
    $var =& $GLOBALS["baz"];
}
foo($bar);
```

Lo que ocurrirá aquí es que *$var* en *foo* será 'ligada' con *$bar* al momento de llamar a la función. Pero luego será 're-ligada' con *$GLOBALS["baz"]*. No existe manera de ligar *$bar* en el ámbito global del script con alguna otra cosa utilizando el mecanismo de Referencias, ya que *$bar* no existe dentro de *foo* (está representado por *$var*, pero *$var* solo está ligado por el contenido, no por el nombre en la tabla de símbolos).

Paso de variables por Referencia

Podemos pasar variables a una función por referencia, para que ésta pueda modificar sus argumentos. La sintáxis es la siguiente :

```
function foo (&$var)
{
    $var++;
}

$a=5;
foo ($a);
// $a ser&aacute; 6 aqui
```

Notar que no hay signo de referencia en la llamada a la función - solo en la definición de la misma. Colocar el signo de referencia solo en la definición de la función alcanza para pasar correctamente el argumento por referencia.

La lista siguiente indica que puede ser pasado por referencia:

- Variables, por ejemplo *foo($a)*
- Operador New, por ejemplo *foo(new foobar())*
- Referencias, devueltas por una función:

```
function &bar()
{
    $a = 5;
    return $a;
}
foo(bar());
```

Se recomienda leer también la explicación sobre retorno por referencia.

Cualquier otro tipo de expresión no debería pasarse por referencia, ya que el resultado sería indefinido. Los ejemplos de paso por referencia siguientes son inválidos:

```
function bar() // Notar que falta &
{
    $a = 5;
    return $a;
}
foo(bar());

foo($a = 5) // Expresi&oacute;n, no variable
foo(5) // Constante, no variable
```

Estos requerimientos son para PHP 4.0.4 y superior.

Retorno por Referencia

Devolver por Referencia es muy útil cuando se quiere utilizar una función para averiguar a que variable debe estar una referencia ligada. Cuando se devuelve por referencia, se debe utilizar esta sintáxis:

```
function &encontrar_var ($param)
{
    ...codigo...
    return $var_encontrada;
}

$foo =& encontrar_var ($bar);
$foo->x = 2;
```

En este ejemplo, el atributo del objeto devuelto por la función *encontrar_var* fue asignado, no ya en la copia, como habría sucedido si no se utilizaba la sintáxis de referencias.

> **Nota:** A diferencia del paso de parámetros, aquí se debe utilizar *&* en ambos lugares - para indicar que se pretende devolver por referencia (y no una copia, como usualmente sucede) y que además esa referencia sea 'ligada' a una variable, y no solo asignada.

Borrando Referencias

Cuando de borra una referencia, solo se rompe esa unión entre el nombre de la variable y el contenido. Esto no significa que el contenido haya sido destruido. Por ejemplo :

```
$a = 1;
$b =& $a;
unset ($a);
```

no destruiría *$b*, solo *$a*.

Nuevamente, sería útil pensar el tema de las referencias como una analogía al comando **unlink** de Unix.

Más Referencias

Muchas construcciones sintácticas en PHP son implementadas utilizando el mecanismo referencial, de manera que todo lo dicho anteriormente sobre las referencias también se aplica a estas construcciones. Algunas de ellas, como pasar o retornar por referencia, se mencionaron antes. Otras construcciones que utilizan referencias son:

Referencias *globales*

Cuando declaramos una variable como **global $var** en realidad estamos creando una referencia a una variable global. Esto significa, que es lo mismo que hacer :

```
$var =& $GLOBALS["var"];
```

De esta manera, por ejemplo, si borramos *$var* no estaríamos borrando la variable global.

$this

En un método de objeto, *$this* es siempre una referencia al objeto que contiene el método.

IV. Seguridad

Tabla de contenidos

Capítulo 22. Introducción

PHP es un poderoso lenguaje e intérprete, ya sea incluido como parte de un servidor web en forma de módulo o ejecutado como un binario CGI separado, es capaz de acceder a archivos, ejecutar comandos y abrir conexiones de red en el servidor. Estas propiedades hacen que cualquier cosa que sea ejecutada en un servidor web sea insegura por naturaleza. PHP está diseñado específicamente para ser un lenguaje más seguro para escribir programas CGI que Perl o C, y con la selección

correcta de opciones de configuración en tiempos de compilación y ejecución, y siguiendo algunas prácticas correctas de programación, PHP le puede dar la combinación precisa de libertad y seguridad que usted necesita.

Ya que hay muchas maneras de utilizar PHP, existen varias opciones de configuración diseñadas para controlar su comportamiento. Un amplio rango de opciones le garantizan que pueda usar PHP para muchos propósitos distintos, pero también quiere decir que hay combinaciones de éstas opciones y configuraciones de servidor que pueden resultar en un entorno inseguro.

El nivel de flexibilidad en la configuración de PHP se compara quizás solo con su flexibilidad de desarrollo. PHP puede ser usado para escribir aplicaciones completas de servidor, con todo el poder de un usuario de un intérprete de comandos, o puede ser usado para inclusiones simples del lado del servidor con muy poco riesgo en un entorno minuciosamente controlado. Cómo construir ese entorno, y qué tan seguro es, básicamente depende del desarrollador PHP.

Este capítulo comienza con algunas recomendaciones generales de seguridad, explica las diferentes combinaciones de opciones de configuración y las situaciones en las que pueden ser usadas con seguridad, y describe diferentes consideraciones a la hora de programar para diferentes niveles de seguridad.

Capítulo 23. Consideraciones generales

Un sistema completamente seguro es prácticamente un imposible, de modo que el enfoque usado con mayor frecuencia en la profesión de seguridad es uno que busque el balance adecuado entre riesgo y funcionalidad. Si cada variable enviada por un usuario requiriera de dos formas de validación biométrica (como rastreo de retinas y análisis dactilar), usted contaría con un nivel extremadamente alto de confiabilidad. También implicaría que llenar los datos de un formulario razonablemente complejo podría tomar media hora, cosa que podría incentivar a los usuarios a buscar métodos para esquivar los mecanismos de seguridad.

La mejor seguridad con frecuencia es lo suficientemente razonable como para suplir los requerimientos dados sin prevenir que el usuario realice su labor de forma natural, y sin sobrecargar al autor del código con una complejidad excesiva. De hecho, algunos ataques de seguridad son simples recursos que aprovechan las vulnerabilidades de este tipo de seguridad sobrecargada, que tiende a erosionarse con el tiempo.

Una frase que vale la pena recordar: Un sistema es apenas tan bueno como el eslabón más débil de una cadena. Si todas las transacciones son registradas copiosamente basándose en la fecha/hora, ubicación, tipo de transacción, etc. pero la verificación del usuario se realiza únicamente mediante una cookie sencilla, la validez de atar a los usuarios al registro de transacciones es mermada severamente.

Cuando realice pruebas, tenga en mente que no será capaz de probar todas las diferentes posibilidades, incluso para las páginas más simples. Los datos de entrada que usted puede esperar en sus aplicaciones no necesariamente tendrán relación alguna con el tipo de información que podría ingresar un empleado disgustado, un cracker con meses de tiempo entre sus manos, o un gato doméstico caminando sobre el teclado. Es por esto que es mejor observar el código desde una perspectiva lógica, para determinar en dónde podrían introducirse datos inesperados, y luego hacer un seguimiento de cómo esta información es modificada, reducida o amplificada.

Internet está repleto de personas que tratan de crearse fama al romper la seguridad de su código, bloquear su sitio, publicar contenido inapropiado, y por lo demás haciendo que sus días sean más

interesantes. No importa si usted administra un sitio pequeño o grande, usted es un objetivo por el simple hecho de estar en línea, por tener un servidor al cual es posible conectarse. Muchas aplicaciones de cracking no hacen distinciones por tamaños, simplemente recorren bloques masivos de direcciones IP en busca de víctimas. Trate de no convertirse en una.

Capítulo 24. Instalación como un binario CGI

Posibles ataques

El uso de PHP como un binario CGI es una opción para el tipo de situaciones en las que por alguna razón no se desea integrar PHP como módulo de algún software de servidor web (como Apache), o en donde se espera usar PHP con diferentes tipos de capas que envuelven el entorno CGI para crear ambientes chroot y setuid seguros para la ejecución de scripts. Esta configuración usualmente involucra la instalación de un binario ejecutable del intérprete PHP en el directorio cgi-bin del servidor web. El aviso de seguridad de CERT CA-96.11 recomienda que se evite la colocación de cualquier intérprete bajo cgi-bin. Incluso si el binario PHP puede ser usado como un intérprete independiente, PHP está diseñado para prevenir el tipo de ataques que esta configuración hace posible:

- Acceso a archivos del sistema: `http://mi.servidor/cgi-bin/php?/etc/passwd`

  La información del query en una URL, la cual viene después del signo de interrogación (?), es pasada como argumentos de línea de comandos al intérprete por la interfaz CGI. Usualmente los intérpretes abren y ejecutan el archivo especificado como primer argumento de la línea de comandos.

  Cuando es invocado como un binario CGI, PHP se rehúsa a interpretar los argumentos de la línea de comandos.

- Acceso a cualquier documento web en el servidor: `http://mi.servidor/cgi-bin/php/zona_secreta/doc.html`

  El segmento de la URL que sigue al nombre del binario de PHP, que contiene la información sobre la ruta `/zona_secreta/doc.html` es usada convencionalmente para especificar el nombre de un archivo que ha de ser abierto e interpretado por el programa CGI. Usualmente, algunas directivas de configuración del servidor web (Apache: Action) son usadas para redireccionar peticiones de documentos como `http://mi.servidor/zona_secreta/script.php` al intérprete de PHP. Bajo este modelo, el servidor web revisa primero los permisos de acceso al directorio `/zona_secreta`, y después de eso crea la petición de redireccionamiento a `http://mi.servidor/cgi-bin/php/zona_secreta/script.php`. Desafortunadamente, si la petición se hace originalmente en esta forma, no se realizan chequeos de acceso por parte del servidor web para el archivo `/zona_secreta/script.php`, únicamente para el archivo `/cgi-bin/php`. De este modo, cualquier usuario capaz de acceder a `/cgi-bin/php` es capaz también de acceder a cualquier documento protegido en el servidor web.

  En PHP, la configuración de tiempo de compilación --enable-force-cgi-redirect y las directivas de configuración en tiempo de ejecución doc_root y user_dir pueden ser usadas para prevenir este tipo de ataques, si el árbol de documentos del servidor llegara a tener

directorio alguno con restricciones de acceso. Consulte las siguientes secciones para una explicación detallada de las diferentes combinaciones.

Caso 1: sólo se sirven archivos públicos

Si su servidor no tiene contenido alguno que no esté restringido por contraseñas o control de acceso basado en direcciones ip, no hay ninguna necesidad de recurrir a estas opciones de configuración. Si su servidor web no le permite hacer redireccionamientos, o el servidor no tiene una forma de comunicarle al binario PHP que la petición de redireccionamiento es segura, puede especificar la opción --enable-force-cgi-redirect en el script de configuración. Aun así debe asegurarse de que sus scripts PHP no dependan de alguna forma especial de hacer llamados al script, ya sea directamente mediante `http://mi.servidor/cgi-bin/php/dir/script.php` ni por la redirección `http://mi.servidor/dir/script.php`.

Los redireccionamientos pueden ser configurados en Apache mediante el uso de directivas AddHandler y Action (vea más adelante).

Caso 2: uso de --enable-force-cgi-redirect

Esta opción en tiempo de compilación previene que cualquier persona haga llamados a PHP directamente mediante una URL como `http://mi.servidor/cgi-bin/php/directorio_secreto/script.php`. En lugar de esto, PHP analizará documentos de esta forma únicamente si han pasado por una regla de redirección del servidor web.

Por lo general, el redireccionamiento en la configuración de Apache es realizada con alguna de las siguientes directivas:

```
Action php-script /cgi-bin/php
AddHandler php-script .php
```

Esta opción ha sido probada únicamente con el servidor web Apache, y depende de que Apache defina la variable de entorno no-estándar *REDIRECT_STATUS* a la hora de gestionar peticiones redirigidas. Si su servidor web no dispone de modo alguno de comunicar si la petición es directa o redirigida, no puede usar esta opción y debe recurrir a alguna de las otras formas documentadas aquí de ejecutar la versión CGI.

Caso 3: configuración de doc_root o user_dir

Incluir contenido activo en los directorios de documentos del servidor web, como scripts y ejecutables, es considerada en ocasiones una práctica insegura. Si, por algún fallo de configuración, los scripts no llegaran a ser ejecutados sino desplegados como documentos HTML normales, esto podría resultar en la revelación de información crítica como trabajos cubiertos por normas de propiedad intelectual o datos de seguridad como contraseñas. Por lo tanto muchos administradores de sistemas preferirán la configuración de otra estructura de directorios para los scripts que sean asequibles únicamente a través del CGI PHP, y por lo tanto deben ser interpretados siempre y no desplegados directamente.

Así mismo, si el método para asegurarse de que las peticiones no son redireccionadas, tal y como se describió en la sección anterior, no está disponible, es necesario entonces configurar un directorio

raíz (doc_root) de scripts que sea diferente al directorio raíz de documentos web.

Puede definir el directorio raíz para scripts de PHP mediante la directiva de configuración doc_root en el archivo de configuración, o puede darle un valor a la variable de entorno *PHP_DOCUMENT_ROOT*. Si ésta está definida, la versión CGI de PHP construirá siempre el nombre del archivo a abrir con este *doc_root* y la información de la ruta dada en la petición, de modo que puede estar seguro de que ningún script será ejecutado por fuera de este directorio (excepto por aquellos indicados en *user_dir*, como se verá a continuación).

Otra opción que puede ser usada en este caso es user_dir. Cuando user_dir no está definida, lo único que controla la apertura de archivos es *doc_root*. Abrir una URL como `http://mi.servidor/~usuario/doc.php` no resulta en la apertura de un archivo bajo el directorio personal del usuario, sino de un archivo llamado `~usuario/doc.php` bajo la ruta doc_root (así es, un directorio cuyo nombre comienza por el carácter de equivalencia [~]).

Si user_dir está definido como, por ejemplo, `public_php`, una petición como `http://mi.servidor/~usuario/doc.php` abrirá un archivo llamado `doc.php` bajo el directorio con el nombre `public_php` ubicado en el directorio personal del usuario. Si el directorio personal del usuario es `/home/usuario`, el archivo ejecutado es `/home/usuario/public_php/doc.php`.

La expansión del valor de *user_dir* ocurre independientemente del parámetro *doc_root*, de modo que es posible controlar el directorio raíz de los documentos y el acceso a los directorios de los usuarios en forma separada.

Caso 4: intérprete PHP por fuera del árbol web

Una opción bastante segura es colocar el intérprete binario de PHP en alguna parte por fuera del árbol de archivos web. En `/usr/local/bin`, por ejemplo. El único inconveniente real con esta alternativa es que ahora usted tendrá que colocar una línea como esta:

```
#!/usr/local/bin/php
```

al comienzo de cualquier archivo que contenga etiquetas PHP. También tendrá que hacer cada archivo ejecutable. Esto quiere decir que debe tratarlo exactamente igual a como trataría cualquier otro script CGI escrito en Perl o sh o cualquier otro lenguaje de scripting común que usara el mecanismo de escape-shell *#!* para el lanzamiento del intérprete.

Para lograr que PHP gestione correctamente la información de *PATH_INFO* y *PATH_TRANSLATED* con este tipo de configuración, el intérprete PHP debe haber sido compilado con la opción de configuración --enable-discard-path.

Capítulo 25. Instalación como módulo de Apache

Cuando PHP es usado como un módulo de Apache, hereda los permisos del usuario de Apache (generalmente los del usuario "nobody"). Este hecho representa varios impactos sobre la seguridad y las autorizaciones. Por ejemplo, si está usando PHP para acceder a una base de datos, a menos que tal base de datos disponga de un control de acceso propio, usted tendrá que hacer que la base de

datos sea asequible por el usuario "nobody". Esto quiere decir que un script malicioso podría tener acceso y modificar la base de datos, incluso sin un nombre de usuario y contraseña. Es completamente posible que un archivador automatizado de documentos web pudiera toparse con la página web de administración de una base de datos, y eliminar todas sus bases de datos. Usted puede protegerse de este tipo de situaciones mediante mecanismos de autorización de Apache, o puede diseñar su propio modelo de acceso usando LDAP, archivos `.htaccess`, etc. e incluir ese código como parte de sus scripts PHP.

Con frecuencia, una vez la seguridad se ha establecido en un punto en donde el usuario de PHP (en este caso, el usuario de apache) tiene asociada una muy leve capacidad de riesgo, se descubre que PHP se encuentra ahora imposibilitado de escribir archivos en los directorios de los usuarios. O quizás se le haya desprovisto de la capacidad de acceder o modificar bases de datos. Se ha prevenido exitosamente que pudiera escribir tanto archivos buenos como malos, o que pudiera realizar transacciones buenas o malas en la base de datos.

Un error de seguridad cometido con frecuencia en este punto es darle permisos de administrador (root) a apache, o incrementar las habilidades del usuario de apache de alguna otra forma.

Escalar los permisos del usuario de Apache hasta el nivel de administrador es extremadamente peligroso y puede comprometer al sistema entero, así que el uso de entornos sudo, chroot, o cualquier otro mecanismo que sea ejecutado como root no debería ser una opción viable para aquellos que no son profesionales en seguridad.

Existen otras soluciones más simples. Mediante el uso de open_basedir usted puede controlar y restringir aquellos directorios que podrían ser usados por PHP. También puede definir áreas solo-Apache, para restringir todas las actividades basadas en web a archivos que no son de usuarios, o del sistema.

Capítulo 26. Seguridad del sistema de archivos

PHP está sujeto a la seguridad misma de la mayoría de sistemas de servidores en lo que a permisos sobre archivos y directorios se refiere. Esto le permite controlar cuáles archivos en el sistema de archivos pueden ser leídos. Debe tenerse cuidado con aquellos archivos que tengan permisos de lectura globales, para asegurarse de que su contenido es seguro y no represente peligro el que pueda ser leído por todos los usuarios con acceso al sistema de archivos.

Ya que PHP fue diseñado para permitir acceso al nivel de usuarios al sistema de archivos, es completamente posible escribir un script PHP que le permita leer archivos del sistema como /etc/passwd, modificar sus conexiones tipo ethernet, enviar trabajos de impresión masivos, etc. Esto tiene algunas implicaciones obvias, en el sentido en que usted tiene que asegurarse de que los archivos desde lo que lee y hacia los que escribe datos, sean los correctos.

Considere el siguiente script, en donde un usuario indica que quisiera eliminar un archivo ubicado en su directorio personal. Este caso asume que se trata de una situación en donde se usa normalmente una interfaz web que se vale de PHP para la gestión de archivos, así que el usuario de Apache tiene permitido eliminar archivos en los directorios personales de los usuarios.

Ejemplo 26-1. Un chequeo pobre de variables nos lleva a...

```
<?php
// eliminar un archivo del directorio personal del usuario

$nombre_usuario = $_POST['nombre_enviado_por_el_usuario'];
$directorio     = "/home/$nombre_usuario";

$archivo_a_eliminar = "$archivo_de_usuario";

unlink ("$directorio/$archivo_de_usuario");

echo "&iexcl;El archivo $archivo_a_eliminar ha sido eliminado!";
?>
```

Ya que el nombre de usuario es enviado desde un formulario de usuario, cualquiera puede enviar un nombre de usuario y archivo propiedad de otra persona, y eliminar archivos. En este caso, usted querrá usar otro método de autenticación. Considere lo que sucede si las variables enviadas son "../etc/" y "passwd". El código entonces se ejecutaría efectivamente como:

Ejemplo 26-2. ... un ataque al sistema de archivos

```
<?php
// elimina un archivo de cualquier parte del disco duro al que el
// usuario de PHP tiene acceso. Si PHP tiene acceso de root:

$nombre_usuario = "../etc/";
$directorio     = "/home/../etc/";

$archivo_a_eliminar = "passwd";

unlink ("/home/../etc/passwd");

echo "&iexcl;El archivo /home/../etc/passwd ha sido eliminado!";
?>
```

Hay dos importantes medidas que usted debe tomar para prevenir estas situaciones.

- Otorgarle únicamente permisos limitados al usuario web del binario PHP.
- Chequear todas las variables que son enviadas por usuarios.

Aquí hay una versión mejorada del script:

Ejemplo 26-3. Un chequeo de nombres de archivos más seguro

```
<?php
// elimina un archivo de cualquier parte del disco duro al que el
// usuario de PHP tiene acceso.

$nombre_usuario = $_SERVER['REMOTE_USER']; // uso de un mecanismo de
                                           // autenticacion

$directorio = "/home/$nombre_usuario";

$archivo_a_eliminar = basename("$archivo_de_usuario"); // remover rutas
unlink ($directorio/$archivo_a_eliminar);

$fp = fopen("/home/registros/eliminacion.log","+a"); // registrar el proceso

$cadena_de_registro = "$nombre_usuario $directorio $archivo_a_eliminar";

fwrite ($fp, $cadena_de_registro);
fclose($fp);

echo "&iexcl;El archivo $archivo_a_eliminar ha sido eliminado!";
?>
```

Sin embargo, incluso este caso no está libre de problemas. Si su sistema de autenticación le ha permitido a los usuarios la creación de sus propios nombres en el sistema, y un usuario elige "../etc/", el sistema se encuenrta nuevamente expuesto. Por esta razón, puede que uster prefiera escribir un chequeo más personalizado:

Ejemplo 26-4. Chequeo de nombres de archivos aun más seguro

```
<?php
$nombre_usuario = $_SERVER['REMOTE_USER']; // uso de un mecanismo de
                                           // autenticacion

$directorio = "/home/$nombre_usuario";

if (!ereg('^[^./][^/]*$', $archivo_de_usuario))
     die('nombre de archivo inv&aacute;lido'); // finalizar,
                                               // no ejecutar el proceso

if (!ereg('^[^./][^/]*$', $nombre_usuario))
     die('nombre de archivo inv&aacute;lido'); // finalizar,
                                               // no ejecutar el proceso

//etc...
?>
```

Dependiendo de su sistema operativo, existe una amplia variedad de archivos sobre los que usted debería estar atento, incluyendo las entradas de dispositivos (/dev/ o COM1), archivos de configuración (archivos /etc/ y los archivos .ini), areas conocidas de almacenamiento de datos (/home/, Mis Documentos), etc. Por esta razón, usualmente es más sencillo crear una política en donde se prohíba toda transacción excepto por aquellas que usted permita explícitamente.

Capítulo 27. Seguridad de Bases de Datos

Hoy en día, las bases de datos son componentes cardinales de cualquier aplicación basada en web, permitiendo que los sitios web provean contenido dinámico. Debido a que información considerablemente sensible o secreta puede ser almacenada en una base de datos, usted debe considerar seriamente la protección de sus bases de datos.

Para recuperar o almacenar cualquier información necesita conectarse a la base de datos, enviar una consulta válida, recoger el resultado y cerrar la conexión. Hoy en día, el lenguaje de consultas usado comúnmente en estas interacciones es el Lenguaje de Consultas Estructurado (SQL por sus siglas en Inglés). Puede apreciar cómo un atacante puede intentar acometidas con una consulta SQL.

Como puede suponer, PHP no puede proteger su base de datos por sí solo. Las siguientes secciones están dirigidas a servir de introducción a los conceptos básicos de cómo acceder y manipular bases de datos desde scripts PHP.

Mantenga en mente esta simple regla: protección en profundidad. Entre más acciones tome para incrementar la protección de su base de datos, menor será la probabilidad de que un atacante tenga éxito exponiendo o abusando de cualquier información almacenada. Un buen diseño del esquema de la base de datos y de la aplicación basta para lidiar con sus mayores temores.

Diseño de Bases de Datos

El primer paso siempre es crear la base de datos, a menos que desee usar una creada por alguien más. Cuando una base de datos es creada, ésta es asignada a un dueño, quien ejecutó la sentencia de creación. Usualmente, únicamente el dueño (o un super-usuario) puede hacer cualquier cosa con los objetos de esa base de datos, y para que otros usuarios puedan usarla, deben otorgarse privilegios.

Las aplicaciones nunca deberían conectarse a la base de datos bajo el usuario correspondiente a su dueño, o como un super-usuario, ya que éstos usuarios pueden, por ejemplo, ejecutar cualquier

consulta a su antojo, modificando el esquema (p. ej. eliminando tablas) o borrando su contenido completo.

Usted puede crear diferentes usuarios de la base de datos para cada aspecto de su aplicación con derechos muy limitados sobre los objetos de la base de datos. Tan solo deben otorgarse los privilegios estrictamente necesarios, y evitar que el mismo usuario pueda interactuar con la base de datos en diferentes casos de uso. Esto quiere decir que si un intruso gana acceso a su base de datos usando las credenciales de sus aplicaciones, él solo puede efectuar tantos cambios como su aplicación se lo permita.

Es buena idea que no implemente toda la lógica del asunto en la aplicación web (es decir, en su script); en su lugar, hágalo en el esquema de la base de datos usando vistas, disparadores o reglas. Si el sistema evoluciona, se espera que nuevos puertos sean abiertos a la aplicación, y tendrá que re-implementar la lógica para cada cliente de la base de datos. Por sobre todo, los disparadores pueden ser usados para gestionar de forma transparente todos los campos automáticamente, lo cual con frecuencia provee información útil cuando se depuren problemas de su aplicación, o se realicen rastreos sobre transacciones particulares.

Conexión con la Base de Datos

Puede que desee establecer las conexiones sobre SSL para encriptar las comunicaciones cliente/servidor, incrementando el nivel de seguridad, o puede hacer uso de ssh para encriptar la conexión de red entre los clientes y el servidor de la base de datos. Si cualquiera de estos recursos es usado, entonces monitorear su tráfico y adquirir información sobre su base de datos será difícil para un atacante potencial.

Modelo de Almacenamiento Encriptado

SSL/SSH protege los datos que viajan desde el cliente al servidor, SSL/SSH no protege los datos persistentes almacenados en la base de datos. SSL es un protocolo sobre-el-cable.

Una vez el atacante adquiere acceso directo a su base de datos (evitando el paso por el servidor web), los datos críticos almacenados pueden estar expuestos o malutilizados, a menos que la información esté protegida en la base de datos misma. La encriptación de datos es una buena forma de mitigar esta amenaza, pero muy pocas bases de datos ofrecen este tipo de mecanismo de encriptación de datos.

La forma más sencilla de evitar este problema es crear primero su propio paquete de encriptación, y luego utilizarlo desde sus scripts de PHP. PHP puede ayudarle en este sentido con varias extensiones, como Mcrypt y Mhash, las cuales cubren una amplia variedad de algoritmos de encriptación. El script encripta los datos antes de insertarlos en la base de datos, y los decripta cuando los recupera. Vea las referencias para consultar más ejemplos de cómo opera la encriptación.

En el caso de datos realmente escondidos, si su representación original no se necesita (es decir, no debe ser desplegada), los resúmenes criptográficos pueden llegar a considerarse también. El ejemplo clásico de gestión de resúmenes criptográficos es el almacenamiento de secuencias MD5 de una contraseña en una base de datos, en lugar de la contraseña misma. Vea también **crypt()** y **md5()**.

Ejemplo 27-1. Uso de un campo de contraseñas encriptado

```
<?php

// almacenamiento de resumen criptografico de la contrasenya

$consulta  = sprintf("INSERT INTO usuarios(nombre,contr) VALUES('%s','%s');",
                     addslashes($nombre_usuario), md5($contrasenya));
$resultado = pg_exec($conexion, $consulta);


// consulta de verificacion de la contrasenya enviada

$consulta  = sprintf("SELECT 1 FROM usuarios WHERE nombre='%s' AND contr='%s';",
                     addslashes($nombre_usuario), md5($contrasenya));
$resultado = pg_exec($conexion, $consulta);

if (pg_numrows($resultado) > 0) {
    echo "&iexcl;Bienvenido, $nombre_usuario!";
}
else {
    echo "No pudo autenticarse a $nombre_usuario.";
}

?>
```

Inyección de SQL

Muchos desarrolladores web no son conscientes de cómo pueden manipularse las consultas SQL, y asumen que una consulta SQL es un comando confiable. Esto representa que las consultas SQL pueden burlar los controles de acceso, y de este modo evitar los chequeos estándares de autenticación y autorización, y a veces las consultas SQL pueden incluso permitir acceso a comandos al nivel del sistema operativo de la máquina huésped.

La Inyección Directa de Comandos SQL es una técnica en la cual un atacante crea o altera comandos SQL existentes para exponer datos escondidos, o sobrescribir datos críticos, o incluso ejecutar comandos del sistema peligrosos en la máquina en donde se encuentra la base de datos. Esto se consigue cuando la aplicación toma información de entrada del usuario y la combina con parámetros estáticos para construir una consulta SQL. Los siguientes ejemplos, desafortunadamente, están basados en historias reales.

Debido a la falta de validación de la información de entrada y el establecimiento de conexiones con la base de datos desde un super-usuario o aquel que puede crear usuarios, el atacante podría crear un super-usuario en su base de datos.

Ejemplo 27-2. Paginación del conjunto de resultados ... y creación de super-usuarios (PostgreSQL y MySQL)

```
<?php

$offset   = argv[0]; // atencion, no se valida la entrada!
$consulta = "SELECT id, nombre FROM productos ORDER BY nombre LIMIT 20 " .
            "OFFSET $offset;";

// con PostgreSQL
$resultado = pg_exec($conexion, $consulta);

// con MySQL
$resultado = mysql_query($consulta);

?>
```

Los usuarios normales pulsan sobre los enlaces 'siguiente' y 'anterior', en donde el desplazamiento (*$offset*) se encuentra codificado en la URL. El script espera que el valor entrante *$offset* sea un número decimal. Sin embargo, qué sucede si alguien intenta un ataque añadiendo una forma

codificada (**urlencode()**) de lo siguiente en la URL

```
// en el caso de PostgreSQL
0;
insert into pg_shadow(usename,usesysid,usesuper,usecatupd,passwd)
    select 'crack', usesysid, 't','t','crack'
    from pg_shadow where usename='postgres';
--

// en el caso de MySQL
0;
UPDATE user SET Password=PASSWORD('crack') WHERE user='root';
FLUSH PRIVILEGES;
```

Si esto ocurriera, entonces el script le presentaría un acceso de superusuario al atacante. Note que *0;* es usado para ofrecer un desplazamiento válido a la consulta original y finalizarla.

> **Nota:** Es una técnica común obligar al analizador sintáctico de SQL a que ignore el resto de la consulta escrita por el desarrollador mediante --, el cual es el signo de comentarios en SQL.

Una forma viable de adquirir contraseñas es jugar con las páginas de resultados de búsquedas. Lo único que necesita el atacante es ver si existen variables enviadas por el usuario que sean usadas en sentencias SQL, y que no sean tratadas apropiadamente. Estos filtros pueden ubicarse por lo general previos a cláusulas *WHERE, ORDER BY, LIMIT* y *OFFSET* en sentencias *SELECT* para personalizar la instrucción. Si su base de datos soporta la construcción *UNION*, el atacante puede intentar añadir una consulta completa a la consulta original para generar una lista de contraseñas desde una tabla cualquiera. El uso de campos encriptados de contraseñas es altamente recomendable.

Ejemplo 27-3. Listado de artículos ... y algunas contraseñas (en cualquier base de datos)

```
<?php

$consulta  = "SELECT id, nombre, insertado, tam FROM productos
                  WHERE tam = '$tam'
                  ORDER BY $orden LIMIT $limite, $offset;";
$resultado = odbc_exec($conexion, $consulta);

?>
```

La parte estática de la consulta puede combinarse con otra sentencia *SELECT* la cual revela todas las contraseñas:

```
'
union select '1', concat(uname||'-'||passwd) as name, '1971-01-01', '0' from usertable;
--
```

Si esta consulta (la cual juega con ′ y --) fuera asignada a una de las variables usadas en *$consulta*, la bestia de la consulta habrá despertado.

Las sentencias UPDATE de SQL son también susceptibles a ataque. Éstas consultas también se encuentran amenazadas por un posible acotamiento y adición de una consulta completamente nueva. Pero en este caso el atacante puede amañar la información de una cláusula *SET*. En este caso se requiere contar con cierta información sobre el esquema de la base de datos para poder manipular la consulta satisfactoriamente. Esta información puede ser adquirida mediante el estudio de los nombres de variables de los formularios, o simplemente por fuerza bruta. No existen demasiadas convenciones para nombrar campos de contraseñas o nombres de usuario.

Ejemplo 27-4. De restablecer una contraseña ... a adquirir más privilegios (con cualquier servidor de base de datos)

```
<?php
$consulta = "UPDATE usertable SET pwd='$pwd' WHERE uid='$uid';";
?>
```

Pero un usuario malicioso envía el valor *' or uid like'%admin%';* -- como *$uid* para cambiar la contraseña del administrador, o simplemente establece *$pwd* a *"hehehe', admin='yes', trusted=100 "* (con un espacio al inicio) para adquirir más privilegios. En tal caso, la consulta sería manipulada:

```
<?php

// $uid == ' or uid like'%admin%'; --
$consulta = "UPDATE usertable SET pwd='...' WHERE uid='' or uid like '%admin%'; --";

// $pwd == "hehehe', admin='yes', trusted=100 "
$consulta = "UPDATE usertable SET pwd='hehehe', admin='yes', trusted=100 WHERE ...;"

?>
```

Un horrible ejemplo de cómo puede accederse a comandos del nivel del sistema operativo en algunas máquinas anfitrionas de bases de datos.

Ejemplo 27-5. Ataque al sistema operativo de la máquina anfitriona de la base de datos (MSSQL Server)

```
<?php

$consulta  = "SELECT * FROM productos WHERE id LIKE '%$prod%'";
$resultado = mssql_query($consulta);

?>
```

Si el atacante envía el valor *a%' exec master..xp_cmdshell 'net user test testpass /ADD'* -- a *$prod*, entones la *$consulta* será:

```
<?php

$consulta  = "SELECT * FROM productos
                    WHERE id LIKE '%a%'
                    exec master..xp_cmdshell 'net user test testpass /ADD'--";
$resultado = mssql_query($consulta);

?>
```

MSSQL Server ejecuta sentencias SQL en el lote, incluyendo un comando para agregar un nuevo usuario a la base de datos de cuentas locales. Si esta aplicación estuviera corriendo como *sa* y el servicio MSSQLSERVER está corriendo con los privilegios suficientes, el atacante tendría ahora una cuenta con la que puede acceder a esta máquina.

> **Nota:** Algunos de los ejemplos anteriores están atados a un servidor de base de datos específico. Esto no quiere decir que un ataque similar sea imposible con otros productos. Su base de datos puede ser vulnerable de forma semejante, en alguna otra manera.

Técnicas de protección

Usted puede argumentar con justa razón que el atacante debe poseer cierta cantidad de información sobre el esquema de la base de datos en la mayoría de ejemplos que hemos visto. Tiene razón, pero usted nunca sabe cuándo y cómo puede filtrarse esta información, y si ocurre, su base de datos estará expuesta. Si está usando un paquete de gestión de bases de datos de código abierto, o cuyo código fuente está disponible públicamente, el cual puede pertenecer a algún sistema de administración de contenido o foro, los intrusos pueden producir fácilmente una copia de un trozo de su código. También puede ser un riesgo de seguridad si es un segmento de código pobremente diseñado.

Estos ataques se basan principalmente en la explotación del código que no ha sido escrito pensando

en la seguridad. Nunca confíe en ningún tipo de información de entrada, especialmente aquella que proviene del lado del cliente, aun si lo hace desde una caja de selección, un campo de entrada hidden o una cookie. El primer ejemplo le muestra que una consulta así de descuidada puede causar desastres.

- Nunca se conecte a la base de datos como un super-usuario o como el dueño de la base de datos. Use siempre usuarios personalizados con privilegios muy limitados.

- Revise si la entrada recibida es del tipo apropiado. PHP posee un amplio rango de funciones de validación de datos, desde los más simples encontrados en Funciones sobre variables y en Funciones de tipo de caracter (p. ej. **is_numeric()**, **ctype_digit()** respectivamente) hasta el soporte para Expresiones Regulares compatibles con Perl.

- Si la aplicación espera alguna entrada numérica, considere la verificación de información con **is_numeric()**, o modifique silenciosamente su tipo usando **settype()**, o utilice su representación numérica, dada por **sprintf()**.
  Ejemplo 27-6. Una forma más segura de generar una consulta para paginado
  ```
  <?php

  settype($offset, 'integer');
  $consulta = "SELECT id, nombre FROM productos ORDER BY nombre " .
              "LIMIT 20 OFFSET $offset;";

  // note el simbolo %d en la cadena de formato, usar %s no tendria sentido
  $consulta = sprintf("SELECT id, nombre FROM productos ORDER BY nombre" .
                      "LIMIT 20 OFFSET %d;", $offset);

  ?>
  ```
- Ubique cada entrada del usuario no-numérica que sea pasada a la base de datos entre comillas con **addslashes()** o **addcslashes()**. Vea el primer ejemplo. Como se ve allí, las comillas colocadas en la parte estática de la consulta no son suficientes, y pueden ser manipuladas fácilmente.

- No imprima ninguna información específica sobre la base de datos, especialmente sobre su esquema, ya sea por razones justas o por ñocaciones. Vea también Reporte de Errores y Gestión de Errores y Funciones de Registro.

- Puede usar procedimientos almacenados y cursores previamente definidos para abstraer el acceso a las bases de datos, de modo que los usuarios no tengan acceso directo a las tablas o vistas, aunque esta solución tiene otros impactos.

Además de estas acciones, usted puede beneficiarse del registro explícito de las consultas realizadas, ya sea desde su script o por la base de datos misma, si ésta soporta la gestión de registros. Por supuesto, el registro de acciones no puede prevenir cualquier intento peligroso, pero puede ser útil para rastrear cuáles aplicaciones han sido usadas para violar la seguridad. El registro en sí no es útil; lo es la información que contiene. Por lo general, es mejor contar con más detalles que con menos.

Capítulo 28. Reporte de Errores

Hablando de la seguridad en PHP, hay dos caras en lo que se concierne al reporte de errores. Una es benéfica al incremento de la seguridad, la otra va en dirección de su detrimento.

Una táctica de ataque típica involucra la acumulación de un perfil de datos del sistema alimentándolo con datos inapropiados, y luego chequear los tipos de errores que son devueltos, y

sus contextos. Esto permite que el cracker del sistema pueda adquirir información del servidor, para así determinar posibles debilidades. Por ejemplo, si un atacante ha recogido información sobre una página creada a partir de los datos de un formulario, él podría intentar sobrescribir las variables, o modificarlas:

Ejemplo 28-1. Ataque a Variables con una página HTML personalizada

```
<form method="post" action="destino_del_ataque?username=badfoo&amp;password=badfoo">
<input type="hidden" name="username" value="badfoo" />
<input type="hidden" name="password" value="badfoo" />
</form>
```

Los errores de PHP que son devueltos normalmente pueden ser bastante útiles para un desarrollador que esté tratando de depurar un script, indicando cosas como la función o archivo que falló, el archivo PHP y el número de línea en donde ocurren los fallos. Toda esta es información de la que puede sacarse provecho. No es extraño que un desarrollador php use **show_source()**, **highlight_string()**, o **highlight_file()** como medida de depuración, pero en un sitio en producción, esta acción puede exponer variables ocultas, sintaxis sin chequear, y otra información peligrosa. Algo especialmente peligroso es ejecutar código que proviene de fuentes bien conocidas con gestores de depuración incorporados, o que usan técnicas de depuración comunes. Si el atacante puede determinar qué técnica general está usando, puede intentar un ataque de fuerza bruta sobre una página, enviando varias cadenas comunes de depuración:

Ejemplo 28-2. Explotación de variables comunes de depuración

```
<form method="post" action="destino_del_ataque?errors=Y&amp;showerrors=1&amp;debug=1">
<input type="hidden" name="errors" value="Y" />
<input type="hidden" name="showerrors" value="1" />
<input type="hidden" name="debug" value="1" />
</form>
```

Independientemente del método de gestión de errores, la capacidad de conseguir que un sistema revele sus posibles estados de error representa un camino para darle información al atacante.

Por ejemplo, el estilo mismo de un error de PHP genérico indica que el sistema está ejecutando PHP. Si el atacante estuviera viendo una página .html, y quisiera consultar qué está siendo usado para la generación de ella por detrás (en busca de debilidades conocidas en el sistema), podría determinar que el sistema fue creado usando PHP alimentándolo con información ñocada.

Un error de función puede indicar si el sistema está ejecutando un tipo particular de motor de base de datos, o dar pistas sobre cómo fue programada o diseñada una página web. Esto facilita posteriores investigaciones en determinados puertos abiertos de bases de datos, o en busca de fallos específicos o debilidades en una página web. Al entregar diferentes trozos de datos inválidos al sistema, por ejemplo, un atacante puede determinar el orden de autenticación en un script, (a partir de los números de línea de los errores) así como averiguar sobre vulnerabilidades que pueden aprovecharse en diferentes puntos del script.

Un error del sistema de archivos o en general de PHP puede indicar qué permisos tiene el servidor web, así como la estructura y organización de los archivos en el servidor web. Algún código de gestión de errores escrito por el desarrollador puede agravar este problema, llevando a la fácil explotación de información hasta entonces "escondida".

Existen tres soluciones principales a este problema. La primera es revisar cuidadosamente todas las funciones, y tratar de compensar por la mayoría de errores encontrados. La segunda es deshabilitar el reporte de errores completamente del código que está siendo ejecutado. La tercera es usar las funciones de gestión de errores personalizables de PHP para crear su propio gestor de errores. Dependiendo de su política de seguridad, puede encontrar que todas ellas pueden ser aplicables a su situación.

Una forma de detectar este problema por adelantado es hacer uso del reporte de errores propio de PHP (**error_reporting()**), para ayudarle a asegurar su código y encontrar uso de variables que

pueda ser peligroso. Al probar su código, previamente a su entrega final, con E_ALL, puede encontrar rápidamente áreas en donde sus variables pueden estar abiertas a la manipulación y explotación en distintas formas. Una vez esté listo para liberar su código, usando E_NONE puede aislarlo de posibles intentos por adquirir información útil para un atacante.

Ejemplo 28-3. Detección de variables peligrosas con E_ALL

```
<?php
if ($nombre_usuario) {  // Variable no inicializada o chequeada antes de su uso
    $login_correcto = 1;
}
if ($login_correcto == 1) { // Si la condicion anterior falla, esta variable
                            // no se encuentra inicializada ni validada
                            // antes de su uso

    readfile ("/informacion/altamente/confidencial/index.html");
}
?>
```

Capítulo 29. Uso de Register Globals

Quizás el cambio más controversial en la historia de PHP se ha dado cuando la directiva register_globals pasó de tener como valor por defecto ON al valor OFF en PHP 4.2.0. La dependencia sobre esta directiva era bastante común y muchas personas nisiquiera estaban enteradas de que existía y asumían que ese era el modo en que PHP trabajaba. Esta página explicará cómo puede llegar a escribirse código inseguro con esta directiva pero tenga en mente que no es la directiva misma la que es insegura sino el uso inapropiado de ella.

Cuando se encuentra activa, la directiva register_globals inyectará (o envenenará) sus scripts con todo tipo de variables, como variables de peticiones provenientes de formularios HTML. Esto junto con el hecho de que PHP no requiere la inicialización de variables significa que es muy fácil escribir código inseguro. Fue una decisión difícil, pero la comunidad de PHP decidió desahibilar esta directiva por defecto. Cuando está habilitada, las personas usan variables sin saber con seguridad de dónde provienen y solo queda asumir. Las variables internas que son definidas en el script mismo son mezcladas con los datos enviados por los usuarios y al deshabilitar register_globals se modifica este comportamiento. Demostremos este caso con un ejemplo del uso incorrecto de register_globals:

Ejemplo 29-1. Ejemplo del uso inapropiado de register_globals = on

```
<?php
// definir $autorizado = true solo si el usuario ha sido autenticado

if (usuario_autenticado()) {
    $autorizado = true;
}

// Ya que no inicializamos $autorizado como false, esta podria estar
// definida a traves de register_globals, como en el caso de GET
// auth.php?autorizado=1

// De modo que cualquier persona podria verse como autenticada!

if ($autorizado) {
    include "/datos/muy/importantes.php";
}
?>
```

Cuando register_globals = on, nuestra lógica anterior podría verse comprometida. Cuando la directiva está deshabilitada, *$autorizado* no puede definirse a través de peticiones, así que no habrá ningún problema, aunque es cierto que siempre es una buena práctica de programación inicializar las variables primero. Por ejemplo, en nuestro ejemplo anterior pudimos haber realizado primero algo como *$authorized = false*. Hacer esto representa que el código anterior podría funcionar con

register_globals establecido a on u off ya que los usuarios no serían autorizados por defecto.

Otro ejemplo es aquel de las sesiones. Cuando register_globals = on, podríamos usar también *$nombre_usuario* en nuestro siguiente ejemplo, pero nuevamente usted debe notar que *$nombre_usuario* puede provenir de otros medios, como GET (a través de la URL).

Ejemplo 29-2. Ejemplo del uso de sesiones con register_globals on u off

```
<?php
// No sabriamos de donde proviene $nombre_usuario, pero sabemos que
// $_SESSION es para datos de sesion

if (isset($_SESSION['nombre_usuario'])) {

    echo "Hola <b>{$_SESSION['nombre_usuario']}</b>";

} else {

    echo "Hola <b>Invitado</b><br />";
    echo "&iquest;Quisiera iniciar su sesi&oacute;n?";

}
?>
```

Incluso es posible tomar medidas preventivas para advertir cuando se intente falsificar la información. Si usted sabe previamente con exactitud el lugar de donde debería provenir una variable, usted puede chequear si los datos enviados provienen de una fuente inadecuada. Aunque esto no garantiza que la información no haya sido falsificada, esto requiere que un atacante adivine el medio apropiado para falsificar la información. Si no le importa de dónde proviene la información, puede usar *$_REQUEST* ya que allí se incluye una mezcla de variables que provienen de datos GET, POST y COOKIE. Consulte también la sección del manual sobre el uso de variables desde fuera de PHP.

Ejemplo 29-3. Detección de envenenamiento simple de variables

```
<?php
if (isset($_COOKIE['COOKIE_MAGICA'])) {

    // COOKIE_MAGICA proviene de una cookie.
    // Asegurese de validar los datos de la cookie!

} elseif (isset($_GET['COOKIE_MAGICA']) || isset($_POST['COOKIE_MAGICA'])) {

   mail("admin@ejemplo.com", "Posible intento de intromision",
        $_SERVER['REMOTE_ADDR']);
   echo "Violaci&oacute;n de seguridad, el administrador ha sido alertado.";
   exit;

} else {

   // COOKIE_MAGICA no fue definida en este REQUEST

}
?>
```

Por supuesto, deshabilitar register_globals no quiere decir que su código vaya a ser seguro. Por cada trozo de datos que sea enviado por el usuario, éste debe ser chequeado en otras formas. ¡Siempre valide los datos de los usuarios e inicialice sus variables! Para chequear por variables no inicializadas, usted puede usar **error_reporting()** para mostrar errores del nivel **E_NOTICE**.

> **Superglobals: Nota de disponibilidad:** Desde 4.1.0, están disponibles algunas matrices superglobales tales como *$_GET*, *$_POST*, y *$_SERVER*, etc. Para más información puede consultar la sección superglobals

---

Capítulo 30. Datos Enviados por el Usuario

Las mayores debilidades de muchos programas PHP no son inherentes al lenguaje mismo, sino simplemente un problema generado cuando se escribe código sin pensar en la seguridad. Por esta razón, usted debería tomarse siempre el tiempo para considerar las implicaciones de cada pedazo de código, para averiguar el posible peligro involucrado cuando una variable inesperada es enviada.

Ejemplo 30-1. Uso Peligroso de Variables

```
<?php
// eliminar un archivo del directorio personal del usuario .. o
// quizas de alguien mas?

unlink ($variable_malvada);

// Imprimir el registro del acceso... o quizas una entrada de /etc/passwd?
fwrite ($desc_archivo, $variable_malvada);

// Ejecutar algo trivial.. o rm -rf *?
system ($variable_malvada);
exec ($variable_malvada);

?>
```

Usted debería examinar siempre, y cuidadosamente su código para asegurarse de que cualquier variable siendo enviada desde un navegador web sea chequeada apropiadamente, y preguntarse a sí mismo:

- ¿Este script afectará únicamente los archivos que se pretende?
- ¿Puede tomarse acción sobre datos inusuales o indeseados?
- ¿Puede ser usado este script en formas malintencionadas?
- ¿Puede ser usado en conjunto con otros scripts en forma negativa?
- ¿Serán adecuadamente registradas las transacciones?

Al preguntarse adecuadamente estas preguntas mientras escribe su script, en lugar de hacerlo posteriormente, usted previene una desafortunada re-implementación del programa cuando desee incrementar el nivel de seguridad. Al comenzar con esta mentalidad, no garantizará la seguridad de su sistema, pero puede ayudar a mejorarla.

Puede que también desee considerar la deshabilitación de register_globals, magic_quotes, u otros parámetros convenientes que pueden causar confusión sobre la validez, fuente o valor de una determinada variable. Trabajar con PHP en modo error_reporting(E_ALL) también puede ayudarle a advertir variables que están siendo usadas antes de ser chequeadas o inicializadas (de modo que puede prevenir que datos inusuales produzcan operaciones inadvertidas).

Capítulo 31. Magic Quotes

Magic Quotes is a process that automagically escapes incoming data to the PHP script. It's preferred to code with magic quotes off and to instead escape the data at runtime, as needed.

What are Magic Quotes

When on, all ′ (single-quote), ″ (double quote), \ (backslash) and *NULL* characters are escaped with a backslash automatically. This is identical to what **addslashes()** does.

There are three magic quote directives:

- magic_quotes_gpc

  Affects HTTP Request data (GET, POST, and COOKIE). Cannot be set at runtime, and defaults to *on* in PHP.

  See also **get_magic_quotes_gpc()**.

- magic_quotes_runtime

  If enabled, most functions that return data from an external source, including databases and text files, will have quotes escaped with a backslash. Can be set at runtime, and defaults to *on* in PHP.

  See also **set_magic_quotes_runtime()** and **get_magic_quotes_runtime()**.

- magic_quotes_sybase

  If enabled, a single-quote is escaped with a single-quote instead of a backslash. If on, it completely overrides magic_quotes_gpc. Having both directives enabled means only single quotes are escaped as ″. Double quotes, backslashes and NULL's will remain untouched and unescaped.

  See also **ini_get()** for retrieving its value.

Why use Magic Quotes

- Useful for beginners

  Magic quotes are implemented in PHP to help code written by beginners from being dangerous. Although SQL Injection is still possible with magic quotes on, the risk is reduced.

- Convenience

  For inserting data into a database, magic quotes essentially runs **addslashes()** on all Get, Post, and Cookie data, and does so automagically.

Why not to use Magic Quotes

- Portability

  Assuming it to be on, or off, affects portability. Use **get_magic_quotes_gpc()** to check for

this, and code accordingly.

- Performance

  Because not every piece of escaped data is inserted into a database, there is a performance loss for escaping all this data. Simply calling on the escaping functions (like **addslashes()**) at runtime is more efficient.

  Although `php.ini-dist` enables these directives by default, `php.ini-recommended` disables it. This recommendation is mainly due to performance reasons.

- Inconvenience

  Because not all data needs escaping, it's often annoying to see escaped data where it shouldn't be. For example, emailing from a form, and seeing a bunch of \' within the email. To fix, this may require excessive use of **stripslashes()**.

Disabling Magic Quotes

The magic_quotes_gpc directive may only be disabled at the system level, and not at runtime. In otherwords, use of **ini_set()** is not an option.

Ejemplo 31-1. Disabling magic quotes server side

An example that sets the value of these directives to *Off* in `php.ini`. For additional details, read the manual section titled How to change configuration settings.

```
; Magic quotes
;

; Magic quotes for incoming GET/POST/Cookie data.
magic_quotes_gpc = Off

; Magic quotes for runtime-generated data, e.g. data from SQL, from exec(), etc.
magic_quotes_runtime = Off

; Use Sybase-style magic quotes (escape ' with '' instead of \').
magic_quotes_sybase = Off
```

If access to the server configuration is unavailable, use of `.htaccess` is also an option. For example:

```
php_flag magic_quotes_gpc Off
```

In the interest of writing portable code (code that works in any environment), like if setting at the server level is not possible, here's an example to disable magic_quotes_gpc at runtime. This method is inefficient so it's preferred to instead set the appropriate directives elsewhere.

Ejemplo 31-2. Disabling magic quotes at runtime

```
<?php
if (get_magic_quotes_gpc()) {
    function stripslashes_deep($value)
    {
        $value = is_array($value) ?
                    array_map('stripslashes_deep', $value) :
                    stripslashes($value);

        return $value;
    }

    $_POST = array_map('stripslashes_deep', $_POST);
    $_GET = array_map('stripslashes_deep', $_GET);
    $_COOKIE = array_map('stripslashes_deep', $_COOKIE);
}
?>
```

Capítulo 32. Ocultando PHP

En general, la seguridad por oscuridad es una de las formas más débiles de seguridad. Pero, en algunos casos, cada pequeño elemento extra de seguridad es deseable.

Unas cuantas técnicas simples pueden ayudarle a esconder PHP, posiblemente retrasando a un atacante que esté intentando descubrir debilidades en su sistema. Al establecer expose_php = off en su archivo `php.ini`, usted reduce la cantidad de información disponible a posibles atacantes.

Otra táctica consiste en configurar los servidores web como apache para que procesen diferentes tipos de archivos como scripts de PHP, ya sea con una directiva `.htaccess`, o en el archivo de configuración de apache mismo. En ese caso puede usar extensiones de archivo que produzcan confusión:

Ejemplo 32-1. Ocultando PHP como otro lenguaje

```
# Hacer que el codigo PHP parezca como otro tipo de codigo
AddType application/x-httpd-php .asp .py .pl
```

U ocultarlo completamente:

Ejemplo 32-2. Uso de tipos desconocidos como extensiones para PHP

```
# Hacer que el codigo PHP parezca como de tipos desconocidos
AddType application/x-httpd-php .bop .foo .133t
```

O escóndalo como código HTML, lo que tiene un pequeño impacto de rendimiento ya que todos los documentos HTML serán procesados por el motor de PHP:

Ejemplo 32-3. Uso de tipos HTML para extensiones PHP

```
# Hacer que todo el codigo PHP luzca como HTML
AddType application/x-httpd-php .htm .html
```

Para que esto funcione de manera efectiva, usted debe renombrar sus archivos PHP con las extensiones anteriores. Aunque es una forma de seguridad por oscuridad, representa una medida preventiva menor con pocos inconvenientes.

Capítulo 33. Mantenerse al Día

PHP, como cualquier otro sistema de tamaño considerable, está bajo constante escrutinio y remodelación. Cada nueva versión incluye con frecuencia cambios mayores y menores para mejorar y reparar fallos de seguridad, problemas de configuración, y otros asuntos que puedan afectar la seguridad y estabilidad global de su sistema.

Como cualquier lenguaje y programa de scripting del nivel del sistema, el mejor enfoque es el de actualizar con frecuencia, y mantenerse alerta sobre las últimas versiones y sus cambios.

V. Características

Tabla de contenidos

Capítulo 34. Autentificación HTTP con PHP

Las caracteríticas de autentificación HTTP en PHP solo están disponibles cuando se está ejecutando como un módulo en Apache y hasta ahora no lo estan en la versión CGI. En un script PHP como módulo de Apache, se puede usar la función **header()** para enviar un mensaje de "Autentificación requerida" al navegador cliente haciendo que muestre una ventana de entrada emergente con nombre de usuario y contraseña. Una vez que el usuario ha rellenado el nombre y la contraseña, la URL que contiene el script PHP será llamada de nuevo con las variables predefinidas *PHP_AUTH_USER*, *PHP_AUTH_PW*, y *AUTH_TYPE* asignadas con el nombre de usuario, la contraseña y el tipo de autentificación respectivamente. Estas variables predefinidas se pueden encontrar en las matrices $_SERVER y *$HTTP_SERVER_VARS*. Sólo autentificación "Básica" está soportada en este momento. Consulte la función **header()** para más información.

> **Nota sobre la versión PHP:** Las Autoglobales, tales como $_SERVER, han estado disponibles desde la versión de PHP 4.1.0. *$HTTP_SERVER_VARS* ha estado disponible desde PHP 3.

Un script de ejemplo que fuerce la autentificación del cliente en una página sería como el siguiente:

Ejemplo 34-1. Ejemplo de autentificación HTTP

```
<?php
  if (!isset($_SERVER['PHP_AUTH_USER'])) {
    header('WWW-Authenticate: Basic realm="My Realm"');
    header('HTTP/1.0 401 Unauthorized');
    echo 'Text to send if user hits Cancel button';
    exit;
  } else {
    echo "<p>Hello {$_SERVER['PHP_AUTH_USER']}.</p>";
    echo "<p>You entered {$_SERVER['PHP_AUTH_PW']} as your password.</p>";
  }
?>
```

> **Nota de compatibilidad:** Por favor tener cuidado cuando esteis programando las lineas de cabecera HTTP. Para garantizar la máxima compatibilidad con todos los clientes, la palabra clave "Basic" debe de ser escrita con "B" mayúscula, la cadena de texto debe estar incluida entre comillas dobles (no simples) y exactamente un espacio debe preceder el código *401* en la linea de cabecera *HTTP/1.0 401*

En vez de, sencillamente, mostrar *PHP_AUTH_USER* y *PHP_AUTH_PW*, seguramente querais comprobar la validez del nombre de usuario y la contraseña. Tal vez enviando una consulta a una base de datos o buscando el usuario en un fichero dbm.

Vigilar aquí los navegadores Interner Explorer con bugs. Parecen muy quisquillosos con el orden de las cabeceras. Enviar la cabecera *WWW-Autentificación* antes que la cabecera HTTP/1.0 401 parece ser el truco por ahora.

En fecha de la versión PHP 4.3.0, para prevenir que alguien escriba un script que revele la contraseña de una página que ha sido autentificada a través de algún mecanismo externo tradicional, las variables PHP_AUTH no serán asignadas si algún tipo de autentificación externa ha sido activada para la página en particular. En este caso, la variable *REMOTE_USER* puede ser usada para identificar al usuario autentificado externamente. Asi que se puedes utilizar *$_SERVER ['REMOTE_USER']*.

> **Configuration Note:** PHP usa la directiva *AuthType* para determinar si una autentificación externa esta en uso.

Nota, a pesar de todo, lo ya dicho no proteje de que alguien que controle una URL no autentificada robe contraseñas de URLs autentificadas en el mismo servidor.

Tanto Netscape como Internet Explorer borrarán la caché de la ventana de autentificación en el navegador local después de recibir una respuesta 401 del servidor. Esto puede usarse, de forma efectiva, para "desconectar" a un usuario, forzandole a reintroducir su nombre y contraseña. Algunas personas usan esto para "hacer caducar" entradas, o para proveer un botón de "desconectar".

Ejemplo 34-2. Ejemplo de autentificación HTTP forzando una reentrada

```
<?php
  function authenticate() {
    header('WWW-Authenticate: Basic realm="Test Authentication System"');
    header('HTTP/1.0 401 Unauthorized');
    echo "You must enter a valid login ID and password to access this resource\n";
    exit;
  }

  if (!isset($_SERVER['PHP_AUTH_USER']) ||
      ($_POST['SeenBefore'] == 1 && $_POST['OldAuth'] == $_SERVER['PHP_AUTH_USER'])) {
   authenticate();
  }
  else {
   echo "<p>Welcome: {$_SERVER['PHP_AUTH_USER']}<br>";
   echo "Old: {$_REQUEST['OldAuth']}";
   echo "<form action='{$_SERVER['PHP_SELF']}' METHOD='POST'>\n";
   echo "<input type='hidden' name='SeenBefore' value='1'>\n";
   echo "<input type='hidden' name='OldAuth' value='{$_SERVER['PHP_AUTH_USER']}'>\n";
   echo "<input type='submit' value='Re Authenticate'>\n";
   echo "</form></p>\n";
  }
?>
```

Este comportamiento no es requerido por el estándar de autentificación básica de HTTP, por lo que nunca debe depender de esto. Pruebas con Lynx han demostrado que Lynx no borra las credenciales de autentificación con una respuesta 401 del servidor, por lo que pulsando atrás y después adelante abriría el recurso de nuevo (siempre que los requerimientos de contraseña no hayan cambiado).

Además tener en cuenta que hasta la version de PHP 4.3.3, la autentificación HTTP no funcionaba con el servidor IIS de Microsoft y la versión CGI de PHP, debido a una limitación de IIS. Para que funcione a partir de PHP 4.3.3, debeis de editar vuestra configuración sobre "Seguridad en directorios" en IIS. Pulsar en 'Editar" y elegir solamente "acceso anonimo", todos los demas campos no se deben de elegir.

Otra limitación es, si estais usando el módulo de IIS (ISAPI), que no podeis usar las variables *PHP_AUTH_**, en su lugar debeis utilizar la variable *HTTP_AUTHORIZATION*. Por ejemplo: *list ($user, $pw) = explode(':', base64_decode(substr($_SERVER['HTTP_AUTHORIZATION'], 6)));*

> **Nota para IIS::** Para que la autentificación HTTP funcione con IIS, la directiva de PHP cgi.rfc2616_headers debe de tener el valor *0* (valor por defecto).
>
> **Nota:** Si safe mode está activado, el uid de el script es agregado a la cabecera *WWW-Authenticate*

Capítulo 35. Cookies

PHP soporta transparentemente cookies HTTP. Las Cookies son un mecanismo que sirve para almacenar datos en el navegador del usuario remoto, para así poder identificar al usuario cuando vuelva. Se pueden poner cookies usando la función **setcookies()**. Las Cookies son parte de la cabecera HTTP, por tanto la función **setcookie()** debe ser llamada antes de que se produzca cualquier salida al navegador. Esta limitación es la misma a la de la función **header()**. Se pueden usar las funciones de almacenamiento intermedio del resultado para retrasar el resultado del script hasta que hayas decidido mandar o no una cookie o cabecera.

Cualquier cookie enviada a ti desde el cliente, automáticamente se convertirá en una variable PHP igual como ocurre con los métodos de datos GET y POST, dependiendo de las variables de configuración *register_globals* y *variables_order*. Si deseas asignar multiples valores a una cookie simple, añade simplemente *[]* a el nombre de la cookie.

En PHP 4.1.0 y posteriores, la matriz auto-global *$_COOKIE* será siempre actualizada con cualquier cookie mandada por el cliente. *$HTTP_COOKIE_VARS* es tambien actualizada en versiones anteriores de PHP cuando la variable de configuración *track_vars* esté activada. (Siempre activada a partir de PHP 4.0.3.)

Para más detalles ver la función **setcookie()**.

Capítulo 36. Sessions

Session support in PHP consists of a way to preserve certain data across subsequent accesses. This enables you to build more customized applications and increase the appeal of your web site. All information is in the Session reference section.

Capítulo 37. Manejo de XForms

XForms define una variación de los tradicionales formularios web que permite que éstos sean usados en una variedad más amplia de plataformas y navegadores, e incluso en medios no-tradicionales como documentos PDF.

La primera diferencia clave en XForms es el modo en que el formulario es enviado al cliente. XForms para Autores HTML contiene una descripción detallada de cómo crear XForms; para los

propósitos de este tutorial, tan solo estaremos viendo un ejemplo sencillo.

Ejemplo 37-1. Un formulario XForms de búsqueda simple

```
<h:html xmlns:h="http://www.w3.org/1999/xhtml"
        xmlns="http://www.w3.org/2002/xforms">
<h:head>
 <h:title>B&uacute;squeda</h:title>
 <model>
  <submission action="http://example.com/buscar"
              method="post" id="s"/>
 </model>
</h:head>
<h:body>
 <h:p>
  <input ref="q"><label>Buscar</label></input>
  <submit submission="s"><label>Iniciar</label></submit>
 </h:p>
</h:body>
</h:html>
```

El anterior formulario despliega una caja de entrada de texto (llamada *q*), y un botón de enviar. Cuando el botón de enviar es pulsado, el formulario será enviado a la página indicada por *action*.

Aquí es en donde empieza a lucir diferente desde el punto de vista de su aplicación web. En un formulario HTML normal, los datos serían enviados como *application/x-www-form-urlencoded*, sin embargo, en el mundo de XForms, esta información es enviada como datos en formato XML.

Si ha tomado la decisión de trabajar con XForms, entonces probablemente quiera los datos en XML, en ese caso, debe echar un vistazo a *$HTTP_RAW_POST_DATA* en donde encontrará el documento XML generado por el navegador, el cual puede pasar a su motor XSLT o intérprete de documentos favorito.

Si no se encuentra interesado en dar formato, y sólo desea que los datos sean cargados en la variable *$_POST* tradicional, puede indicarle al navegador del cliente que envíe sus datos como *application/x-www-form-urlencoded* modificando el atributo *method* al valor *urlencoded-post*.

Ejemplo 37-2. Uso de un XForm para poblar *$_POST*

```
<h:html xmlns:h="http://www.w3.org/1999/xhtml"
        xmlns="http://www.w3.org/2002/xforms">
<h:head>
 <h:title>B&uacute;squeda</h:title>
 <model>
  <submission action="http://example.com/search"
              method="urlencoded-post" id="s"/>
 </model>
</h:head>
<h:body>
 <h:p>
  <input ref="q"><label>Buscar</label></input>
  <submit submission="s"><label>Iniciar</label></submit>
 </h:p>
</h:body>
</h:html>
```

> **Nota:** Al momento en que se escriben estas líneas, muchos navegadores no ofrecen soporte para XForms. Revise la versión de su navegador si el anterior ejemplo falla.

Capítulo 38. Manejo de envío de archivos

Envío de archivos con el método POST

PHP es capaz de recibir envíos de archivo de cualquier navegador que cumpla la norma RFC-1867 (entre los que se incluyen Netscape Navigator 3 o posterior, Microsoft Internet Explorer 3 con un parche o posterior sin éste). Ésta característica permite que los usuarios envien archivos de texto y binarios. Mediante la autentificación y funciones de manejo de archivos de PHP, es posible un control total de quién puede enviar archivos y que se hace con éstos una vez recibidos.

Es importante destacar que PHP también soporta el método PUT para envío de archivos tal y como lo utiliza Netscape Composer y el cliente Amaya de W3C. Consulte Soporte del método PUT para más detalles.

Una página de envío de archivos se puede crear mediante un formulario parecido a éste:

Ejemplo 38-1. Formulario de envío de archivo

```
<form enctype="multipart/form-data" action="_URL_" method="post">
<input type="hidden" name="MAX_FILE_SIZE" value="1000">
Send this file: <input name="userfile" type="file">
<input type="submit" value="Send File">
</form>
```

La _URL_ debe tener como destino un script PHP. El input oculto MAX_FILE_SIZE debe encontrarse antes del input de tipo "file" para indicar el tamaño máximo de archivo que se puede enviar en bytes

Aviso
MAX_FILE_SIZE debe ser consultado por el navegador; aun así es sencillo saltarse este máximo por lo tanto no se debe presuponer que el navegador siempre lo respetará. En contrapartida, la configuracion de PHP relativa al tamaño maximo no puede ser obviada.

Las variables definidas para los archivos enviados varian en función de la versión y configuración de PHP que se utilice. Las variables de las que hablamos a continuación serán definidas en la página destino despues de una recepción de fichero correcta. Cuando track_vars este activado, el array $HTTP_POST_FILES/$_FILES se inicializará. Por ultimo, las variables relacionadas seran inicializadas como globales cuando register_globals esté habilitado. Cabe señalar que el uso de las variables globales no esta recomendado en ningún caso.

> **Nota:** track_vars esta activado siempre desde PHP 4.0.3. A partir de PHP 4.1.0 , $_FILES puede ser utilizado alternativamente a *$HTTP_POST_FILES*. *$_FILES* es siempre global asi que *global* no debe ser usado con $_FILES en el ámbito de función.

$HTTP_POST_FILES*/*$_FILES contienen la información sobre el fichero recibido.

A continuación se describe el contenido de *$HTTP_POST_FILES*. Se ha tomado el nombre 'userfile' para el fichero recibido tal y como se usaba en el script de ejemplo anterior:

$HTTP_POST_FILES['userfile']['name']

> El nombre original del fichero en la máquina cliente.

$HTTP_POST_FILES['userfile']['type']

> El tipo mime del fichero (si el navegador lo proporciona). Un ejemplo podría ser *"image/gif"*.

$HTTP_POST_FILES['userfile']['size']

El tamaño en bytes del fichero recibido.

$HTTP_POST_FILES['userfile']['tmp_name']

El nombre del fichero temporal que se utiliza para almacenar en el servidor el archivo recibido.

Nota: A partir de PHP 4.1.0 se puede utilizar el variable corta *$_FILES*. PHP 3 no soporta *$HTTP_POST_FILES*.

Cuando register_globals se activa en el `php.ini` las siguientes variables son accesibles. Se ha tomado el nombre 'userfile' para el fichero recibido tal y como se usaba en el script de ejemplo del principio:

- *$userfile* - El nombre del fichero temporal que se utiliza para almacenar en el servidor el archivo recibido.
- *$userfile_name* - El nombre original del fichero en la máquina cliente.
- *$userfile_size* - El tamaño en bytes del fichero recibido.
- *$userfile_type* - El tipo mime del fichero (si el navegador lo proporciona). Un ejemplo podría ser "image/gif".

Se puede ver que "*$userfile*" (en las variables anteriores) toma el valor del atributo "name" que contenga el campo <input> de tipo "file" del formulario de envio. En el ejemplo anterior, elegimos llamarlo "userfile".

Nota: *register_globals = On* se desaconseja por razones de seguridad y rendimiento.

Por defecto, los ficheros serán almacenados en el directorio temporal por defecto del servidor a no ser que se especifique otra localizacion con la directiva upload_tmp_dir en `php.ini`. El directorio temporal por defecto del servidor puede ser modificado cambiando el valor de la variable de entorno *TMPDIR* en el contexto en que se ejecuta PHP La configuración de las variables de entorno no se puede realizar en PHP a través de la función **putenv()**. Esta variable de entorio puede ser utilizada también para asegurarnos que otras operaciones con archivos recibidos están funcionando correctamente.

Ejemplo 38-2. Verificando los archivos recibidos

Los siguientes ejemplos son validos para versiones de PHP 4 superiores a la 4.0.2. Veanse las funciones **is_uploaded_file()** y **move_uploaded_file()**.

```
<?php
// In PHP 4.1.0 or later, $_FILES should be used instead of $HTTP_POST_FILES.
if (is_uploaded_file($HTTP_POST_FILES['userfile']['tmp_name'])) {
    copy($HTTP_POST_FILES['userfile']['tmp_name'], "/place/to/put/uploaded/file");
} else {
    echo "Possible file upload attack. Filename: " . $HTTP_POST_FILES['userfile']['name
}
/* ...or... */
move_uploaded_file($HTTP_POST_FILES['userfile']['tmp_name'], "/place/to/put/uploaded/fi
?>
```

El script PHP que recibe el fichero, debe implementar la lógica necesaria para determinar que debe ser realizado con el fichero. Se puede utilizar, por ejemplo, la variable *$HTTP_POST_FILES ['userfile']['size']* para descartar los ficheros demasiado chicos o demasiado grandes; por otro lado,

se puede usar la variable *$HTTP_POST_FILES['userfile']['type']* para descartar los que no se ajusten a algun criterio de tipo. Cualquiera que sea la logica que utilicemos, se debe borrar o mover el archivo del directorio temporal.

El archivo será borrado del directorio temporal al final de la petición si no se ha movido o renombrado.

Errores comunes

A *MAX_FILE_SIZE* no se le puede dar un valor mayor que el valor que se haya especificado en la directivaupload_max_filesize. Por defecto se tiene un límite de 2 MegaBytes.

Si se ha activado el límite de memoria, se deben especificar un valor alto para memory_limit. En cualquier caso, se debe asegurar un valor suficientemente grande para memory_limit.

Si max_execution_time tiene un valor muy pequeño, la ejecución del script puede exceder este valor. De esta forma, se debe asegurar un valor suficientemente grande para *max_execution_time*.

Si post_max_size tiene un valor muy pequeño, los ficheros mas grandes a este valor, no podrán ser enviados. Por ello, se debe asegurar un valor suficientemente grande para *post_max_size*.

No verificar que ficheros se estan manipulando puede tener como consecuencia que los usuarios puedan acceder a información sensible en otros directorios.

Cabe señalar que el httpd de CERN parece cortar todo a partir del primer espacio en blanco en el "content-type" de la cabecera mime que obtiene del cliente. Si este es el caso, con el httpd de CERN no se soporta la funcionalidad de envío de ficheros.

Envío de multiples ficheros

Se pueden enviar multiples ficheros usando diferentes nombres (*name*) para los *input*.

Así mismo, es posible enviar varios archivos simultaneamente y tener organizada en arrays la información. Para hacer esto, se utiliza la misma sintáxis que cuando tenemos multiples "selects" o "checkboxes" en el formulario HTML:

> **Nota:** El soporte para envío multiple de ficheros fue añadido en la versión 3.0.10.

Ejemplo 38-3. Envío de multiples ficheros

```
<form action="file-upload.php" method="post" enctype="multipart/form-data">
  Send these files:<br>
  <input name="userfile[]" type="file"><br>
  <input name="userfile[]" type="file"><br>
  <input type="submit" value="Send files">
</form>
```

Cuando el formulario del ejemplo es enviado, los arrays *$HTTP_POST_FILES['userfile']*, *$HTTP_POST_FILES['userfile']['name']* y *$HTTP_POST_FILES['userfile']['size']* son inicializados. Así mismo pasa con $_FILES en PHP 4.1.0 o superiores y $HTTP_POST_VARS en PHP 3. Cuando *register_globals* esta activa, las variables globales para los archivos recibidos también son inicializadas. Cada uno de estos arrays tendrá en los índices numericos correspondientes los valores para cada fichero recibido.

Por ejemplo, si tomamos como nombres de archivo enviados `/home/test/review.html` y `/home/test/xwp.out`. Tendríamos en *$HTTP_POST_FILES['userfile']['name'][0]* el valor de `review.html`, y en *$HTTP_POST_FILES['userfile']['name'][1]* tendríamos `xwp.out`; analogamente, *$HTTP_POST_FILES['userfile']['size'][0]* contendría el tamaño del fichero `review.html`, y asi sucesivamente...

$HTTP_POST_FILES['userfile']['name'][0], *$HTTP_POST_FILES['userfile']['tmp_name'][0]*, *$HTTP_POST_FILES['userfile']['size'][0]* y *$HTTP_POST_FILES['userfile']['type'][0]* tambien son asignadas.

Soporte del método PUT

PHP soporta el metodo HTTP PUT que usan aplicaciones como Netscape Composer y Amaya del W3C. Las peticiones PUT son más sencillas que el método POST. Un ejemplo:

```
PUT /path/filename.html HTTP/1.1
```

Esto normalmente significaría que el cliente remoto quiere salvar el contenido como: /path/filename.html en tu árbol web. Lógicamente no una buena idea que la gente pueda escribir en tu árbol web. Para manipular esta petición debes decirle al servidor que esta petición sea atendida por un script PHP. En Apache, por ejemplo, se utiliza para esto la directiva *Script* en los alguno de los archivos de configuración del servidor. Un sitio típico de uso es dentro del bloque <Directory> o quizás en el bloque <Virtualhost>. Una linia así deberia hacer ésta función:

```
Script PUT /put.php
```

Ésto le dice a Apache que envíe todas peticiones PUT para URIs que contengan esta linia al script put.php. Se asume que PHP se encuentra activo y con la extensión .php enlazada a él.

Dentro del script put.php3 se podría implementar algo así:

```
<?php copy($PHP_UPLOADED_FILE_NAME,$DOCUMENT_ROOT.$REQUEST_URI); ?>
```

Esto copiaría el archivo a la localización requerida por el cliente remoto. Aqui se pueden ejecutar funciones de autentificación de usuario o cualquier otro tipo de chequeo. El archivo se guarda en el archivo temporal del sistema servidor de la misma manera que el Método POST. Cuando la petición finaliza, el archivo temporal es eliminado. En consequencia el script dede proceder al trato de éste inmediatamente, ya sea para copiarlo, renombrarlo, etc. El archivo se encuentra en la variable $PHP_PUT_FILENAME, y el destino sugerido por el cliente en la variable $REQUEST_URI (puede variar en servidores web que no sean Apache). No es necesario hacer caso al destino sugerido por el cliente. Por ejemplo se podrían copiar los archivos enviados a directorios especialmente designados para esta tarea.

Capítulo 39. Usando archivos remotos

Siempre que *allow_url_fopen* esté habilitado en `php.ini`, se pueden usar URLs HTTP y FTP con la mayoría de las funciones que toman un archivo como parámetro. Además URLs pueden ser usadas con **include()**, **include_once()**, **require()** y **require_once()**. Consultar Apéndice L para más información sobre los protocolos soportados por PHP.

Nota: En PHP 4.0.3 y versiones anteriores, para usar envolturas URL, habia que configurar PHP usando la opción de configuración *--enable-url-fopen-wrapper*.

Nota: Las versiones para windows de PHP anteriores a PHP 4.3 no soportaban acceso remoto a ficheros en las funciones siguientes: **include()**, **include_once()**, **require()**, **require_once()**, y las funciones imagecreatefromXXX de la extensión Referencia LII, *Funciones para imágenes*.

Por ejemplo, se puede usar este para abrir un archivo en un servidor web remoto, analizar en la salida la información que se quiera, y entonces, usar la información en una consulta a base de datos, o simplemente para sacarlas en un estilo que coincida con el resto de su sitio web.

Ejemplo 39-1. Obtener el título de una página remota

```
<?php
$file = fopen ("http://www.example.com/", "r");
if (!$file) {
    echo "<p>Unable to open remote file.\n";
    exit;
}
while (!feof ($file)) {
    $line = fgets ($file, 1024);
    /* This only works if the title and its tags are on one line */
    if (eregi ("<title>(.*)</title>", $line, $out)) {
        $title = $out[1];
        break;
    }
}
fclose($file);
?>
```

También se puede escribir a archivos en un servidor FTP (siempre que se conecte como un usuario con los correctos derechos de acceso). Solamente se pueden crear nuevos ficheros usando este método; si se intenta sobreescribir un fichero ya existente, la función **fopen()** fallará

Para conectar como un usuario distinto de 'anonymous', se necesita especificar el nombre de usuario (y posiblemente contraseña) dentro de la URL, tales como 'ftp://usuario:clave@ftp.ejemplo.com/camino/a/archivo'. (Se puede usar la misma clase de sintaxis para acceder a archivos via HTTP cuando se requería una autenticació de same sort of syntax to access files via HTTP when they require Basic authentication.)

Ejemplo 39-2. Almacenando datos en un servidor remoto

```
<?php
$file = fopen ("ftp://ftp.example.com/incoming/outputfile", "w");
if (!$file) {
    echo "<p>Unable to open remote file for writing.\n";
    exit;
}
/* Write the data here. */
fwrite ($file, $_SERVER['HTTP_USER_AGENT'] . "\n");
fclose ($file);
?>
```

Nota: Podeis creer por el ejemplo anterior, que podeis usar esta tecnica para escribir en un fichero de registro remoto. Desgraciadamente no funcionaria porque la llamada **fopen()** fallaria si el fichero remoto existe. Para usar registros distribuidos de esa manera podeis consultar la funcion **syslog()**.

Capítulo 40. Manejando conexiones

Nota: Todo lo siguiente se aplica a partir de la versión 3.0.7 y posterior.

Internamente en PHP se mantiene el estado de la conexión. Hay 3 posibles estados:

- 0 - NORMAL
- 1 - ABORTED (Abortado)
- 2 - TIMEOUT (Fuera de tiempo)

Cuando un script PHP se está ejecutando se activa el estado NORMAL. Si el cliente remoto se desconecta, se pasa al estado ABORTED. Esto suele ocurrir cuando el usuario pulsa en el botón STOP del navegador. Si se alcanza el límite de tiempo impuesto por PHP (ver **set_time_limit()**), se pasa al estado TIMEOUT.

Puedes decidir si quieres que la desconexión de un cliente cause que tu script sea abortado. Algunas veces es cómodo que tus scripts se ejecuten por completo, incluso si no existe ya un navegador remoto que reciba la salida. El comportamiento por defecto es sin embargo, que tu script se aborte cuando el cliente remoto se desconecta. Este comportamiento puede ser configurado vía la directiva ignore_user_abort en el fichero php3.ini, o también con la función **ignore_user_abort()**. Si no le especificas al PHP que cuando un usuario aborte lo ignore, tu script terminará su ejecución. La única excepción es si tienes registrada un función de desconexión usando la función **register_shutdown_function()**. Con una función de desconexión, cuando un usuario remoto pulsa en el botón STOP, la próxima vez que tu script intenta mostrar algo, PHP detecta que la conexión ha sido abortada y se llama a la función de desconexión. Esta función de desconexión también se llama al final de la ejecución de tu script cuando se ha ejecutado normalmente, de manera que si quieres hacer algo diferente en caso de que un cliente se haya desconectado, puedes usar la función **connection_aborted()**. Esta función devuelve **`TRUE`** si la conexión fue abortada.

Vuestro script también se puede terminar por un temporizador interno. El timeout por defecto es de 30 segundos. Se puede cambiar usando la directiva max_execution_time en el fichero `php.ini` o la correspondiente directiva php_max_execution_time en la configuración del servidor de páginas Apache, como también con la función **set_time_limit()**. Cuando el temporizador expira, el script se aborta como en el caso de la desconexión del cliente, de manera que si se ha definido una función de desconexión, esta se llamará. Dentro de esta función de desconexión, puedes comprobar si fue el timeout el que causó que se llamara a la función de desconexión, llamando a la función **connection_timeout()**. Esta función devolverá verdadero si el timeout causa que se llame a la función de desconexión.

Hay que destacar que ambos, el estado ABORTED y el TIMEOUT, se pueden activar al mismo tiempo. Esto es posible si le dices a PHP que ignore las desconexiones intencionadas de los usuarios. PHP aún notará el hecho de que el usuario puede haberse desconectado, pero el script continuará ejecutándose. Si se alcanza el tiempo límite de ejecución será abortado y, si se ha definido una función de desconexión, esta será llamada. En este punto, encontrarás que las funciones **connection_timeout()** y **connection_aborted()** devuelven verdadero. Puedes comprobar ambos estados de una manera simple usando la función **connection_status()**. Esta función devuelve un campo de bit de los estados activos. De este modo, si ambos estados están activos devolvería por ejemplo un valor 3.

Capítulo 41. Conexiones persistentes a bases de datos

Las conexiones persistentes son enlaces que no se cierran cuando termina la ejecución del archivo de comandos. Cuando se pide una conexión persistente, PHP comprueba si hay ya una conexión persistente idéntica (que permanecía abierta desde antes) - y si existe, la usa. Si no existe, crea un enlace. Una conexión 'idéntica' es una conexión que se abrió hacia el mismo "host", con el mismo nombre de usuario y la misma contraseña (donde sea aplicable).

La gente que no está familiarizada con el modo como trabajan y distribuyen la carga los servidores "web" puede confundir que significa conexiones persistentes. En particular, *no* te dan la habilidad de abrir 'sesiones de usuario' en el mismo enlace, *no* dan la habilidad de construir una transacción de forma eficiente, y no hacen un montón de otras cosas. De hecho, para ser extremadamente claros sobre el tema las conexiones persistentes no te dan *ninguna* functionalidad que no fuera posible con sus hermanas no-persistentes.

¿Por qué?

Esto tiene que ver con el modo como funcionan los servidores "web". Hay tres modos en que un servidor "web" puede utilizar PHP para generar páginas web.

El primer método es usar PHP como una capa CGI. Cuando corre de este modo, se crea y destruye una instancia del intérprete PHP por cada página solicitada (para una página PHP) a tu servidor. Debido a que se destruye después de cada petición, cualquier recurso que adquiera (como un enlace a un servidor de base de datos SQL) se cierra cuando es destruido. En este caso, no se gana nada si se intentan usar conexiones persistentes, ya que simplemente no persisten.

El segundo, y más popular, método es correr PHP como un módulo en un servidor web multiproceso, lo cual actualmente sólo incluye Apache. Un servidor multiproceso tiene típicamente un proceso (el padre) que coordina un conjunto de procesos (sus hijos) que realmente hacen el trabajo se servir las páginas web. Cuando entra cada petición de un cliente, es entregada a uno de los hijos que no esté ya sirviendo a otro cliente. Esto significa que cuando el mismo cliente hace una segunda petción al servidor, puede ser atendido por un proceso hijo distinto del de la primera vez. Lo que una conexión persistente hace por ti en este caso es hacerlo de tal modo que cada proceso hijo sólo necesita conectar a tu SQL server la primera vez que sirve una página que hace uso de una conexión así. Cuando otra página solicita una conexión a SQL server, puede reutilizar la conexión que el hijo estableció previamente.

El último método es usar PHP como un "plug-in" para un servidor web multihilo. En la actualidad es solamente teórico -- PHP no funciona aún como "plug-in" para ningún servidor web multihilo. Actualmente PHP 4 soporta ISAPI, WSAPI y NSAPI (en Windows), lo cual permite a PHP ser utilizado como "plug-in" para servidores web multihilo como Netscape FastTrack, Internet Information Server (IIS) de Microsoft, y O'Reilly's WebSite Pro. El comportamiento es exactamente el mismo que para el modelo de multiprocesador descrito anteriormente. Tener en cuanta que el soporte para SAPI no está disponible en PHP 3.

Si las conexiones persistentes no aportan ninguna funcionalidad añadida, ¿para qué son buenas?

La respuesta aqui es extremadamente simple -- eficiencia. Las conexiones persistentes son buenas si las cabeceras de control para crear un enlace a tu servidor SQL es alta. Que estas cabeceras sean o no realmente altas depende de muchos factores. Como, qué clase de base de datos es, si esta o no situada en el mismo ordenador que el servidor web, cómo está de cargada la máquina donde se

encuentre el servidor SQL, y otras así. El hecho fundamental es que si la cabecera de conexión es alta, las conexiones persistentes te ayudan considerablemente . Ellas hacen que el proceso hijo simplemente conecte solamente una vez durante todo su intervalo de vida, en vez de cada vez que procesa una pagina que requiere conectar al servidor SQL. Esto significa que por cada hijo que abrió una conexión persistente tendrá su propia conexión persistente al servidor. Por ejemplo, si tienes 20 procesos hijos distintos que corran un archivo de comandos que cree una conexión persistente a tu servidor SQL, tendrías 20 conexiones diferentes a ti servidor SQL, una por cada hijo.

No obstante, hay que tener en cuenta que esto puede tener desventajas si estais utilizando una base de datos con límites de conexión, por debajo del numero de procesos hijo con conexiones persistentes. Si tu base de datos tiene un límite de 16 conexiones simultaneas y en el curso de una sesión de servidor, 17 procesos hijo intentan conectarse, uno de ellos no podrá hacerlo. Si existen errores en los scripts, que no permitan terminar la conexion (p.ej.bucles infinitos), una base de datos con solo 16 conexiones puede ser rápidamente hundida. Comprobar la documentacion de vuestra base de datos para obtener información sobre que hacer con conexiones abandonadas ó libres.

Aviso
Un par de advertencias más a tener en cuenta cuando utiliceis conexiones persistentes. La primera, si utilizais bloqueos en una tabla desde una conexión persistente y por cualquier causa el script no puede desbloquear la tabla, todos los scripts posteriores que usen esta conexión, quedarán bloqueados indefinídamente y se requerirá que, ó bien el servidor httpd ó la base de datos sean arrancados de nuevo. La segunda, cuando utiliceis transacciones, un bloqueo por transacción será heredado por el próximo script usando la conexión, si la ejecución del primer script termina antes que el bloqueo. en ambos caso podeis utilizar **register_shutdown_function()** para registrar una funcion simple de limpieza que desbloquee las tablas ó deshaga la transacción. Lo mejor para evitar problemas es no usar conexiones persistentes en scripts que usen bloqueos de tablas ó transacciones (para todolo demás pueden ser usadas sin problemas)

Un resumen importante. Las conexiones persistentes fueron diseñadas para tener una ñalencia uno-a-uno con las conexiones normales. Eso significa que deberíais *siempre* ser capaz de reemplazar las conexiones persistentes por conexiones no persistentes y no cambiará el modo como se comporta el archivo de comandos. *Puede* cambiar la eficiencia del archivo de comandos (y probablemete lo hará), ¡pero no su comportamiento!

Ver tambien **fbsql_pconnect()**, **ibase_pconnect()**, **ifx_pconnect()**, **imap_popen()**, **ingres_pconnect()**, **msql_pconnect()**, **mssql_pconnect()**, **mysql_pconnect()**, **ociplogon()**, **odbc_pconnect()**, **ora_plogon()**, **pfsockopen()**, **pg_pconnect()** y **sybase_pconnect()**.

Capítulo 42. Modo Seguro (Safe Mode)

El Modo Seguro de PHP es un intento para resolver el problema de la seguridad en un servidor compartido. Tratar de resolver este problema al nivel de PHP es arquitectónicamente incorrecto, pero ya que las alternativas en un servidor web y a niveles de sistemas operativos no son tan realistas, mucha gente, especialmente la de proveedores de Internet (ISP), usa el Modo Seguro por ahora.

Tabla 42-1. Las directivas de Configuración que controlan el Modo Seguro son:

Directiva	Valor por Omisión
safe_mode	*Off*
safe_mode_gid	*0*

Directiva	Valor por Omisión
safe_mode_include_dir	""
safe_mode_exec_dir	*1*
open_basedir	""
safe_mode_allowed_env_vars	*PHP_*
safe_mode_protected_env_vars	*LD_LIBRARY_PATH*
disable_functions	""

Cuando safe_mode está en On, el PHP verifica si el dueño del script actual coincide con el dueño del fichero a ser operado por una función de fichero. Por ejemplo:

```
-rw-rw-r--    1 rasmus   rasmus       33 Jul  1 19:20 script.php
-rw-r--r--    1 root     root       1116 May 26 18:01 /etc/passwd
```

Corriendo este script.php

```
<?php
 readfile('/etc/passwd');
?>
```

resulta in este error cuando Modo Seguro está habilitado:

```
Warning: SAFE MODE Restriction in effect. The script whose uid is 500 is not
allowed to access /etc/passwd owned by uid 0 in /docroot/script.php on line 2
```

Sin embargo, pueden haber ambientes donde una estricta verificación del *UID* no es apropiada, y una relajada verificación del *GID* es suficiente. Esto es soportado por medio del switch safe_mode_gid. Seteándolo a *On* hace la verificación relajada *GID*, seteándolo a *Off* (el valor por omisión) hace la verificación del *UID*.

Si en vez del safe_mode, Ud. setea un directorio open_basedir, entonces todas las operaciones de fichero estarán limitadas a los ficheros bajo ese directorio especificado. Por ejemplo (ejemplo de httpd.conf de Apache):

```
<Directory /docroot>
  php_admin_value open_basedir /docroot
</Directory>
```

Si Ud. corre el mismo script.php con este seteo open_basedir, entonces este es el resultado:

```
Warning: open_basedir restriction in effect. File is in wrong directory in
/docroot/script.php on line 2
```

Ud. también puede inhabilitar funciones individuales. Note que la directiva disable_functions no puede ser usada fuera del fichero `php.ini` lo que significa que Ud. no puede inhabilitar funciones en los principios per-virtualhost o per-directory en su fichero httpd.conf. Si agregamos esto a nuestro fichero `php.ini`:

```
disable_functions readfile,system
```

Entonces obtenemos esta salida:

```
Warning: readfile() has been disabled for security reasons in
/docroot/script.php on line 2
```

Funciones restringidas/inhabilitadas por Modo Seguro

Esta es una lista probablemente incompleta y posiblemente incorrecta de las funciones limitadas por safe mode.

Tabla 42-2. Funciones limitadas por Modo Seguro

Función	Limitaciones
dbmopen()	Comprueba que los archivos/directorios que va a utilizar, tengan la misma UID que el script que está siendo ejecutado.
dbase_open()	Comprueba que los archivos/directorios que va a utilizar, tengan la misma UID que el script que está siendo ejecutado.
filepro()	Comprueba que los archivos/directorios que va a utilizar, tengan la misma UID que el script que está siendo ejecutado.
filepro_rowcount()	Comprueba que los archivos/directorios que va a utilizar, tengan la misma UID que el script que está siendo ejecutado.
filepro_retrieve()	Comprueba que los archivos/directorios que va a utilizar, tengan la misma UID que el script que está siendo ejecutado.
ifx_*()	restricciones sql_safe_mode, (!= safe mode)
ingres_*()	restricciones sql_safe_mode, (!= safe mode)
mysql_*()	restricciones sql_safe_mode, (!= safe mode)
pg_loimport()	Comprueba que los archivos/directorios que va a utilizar, tengan la misma UID que el script que está siendo ejecutado.
posix_mkfifo()	Comprueba si el directorio que va a utilizar, tiene la misma UID que el script que está siendo ejecutado.
putenv()	Obecede las ini-directivas safe_mode_protected_env_vars y safe_mode_allowed_env_vars. Vea también la documentación de **putenv()**
move_uploaded_file()	Comprueba que los archivos/directorios que va a utilizar, tengan la misma UID que el script que está siendo ejecutado.
chdir()	Comprueba si el directorio que va a utilizar, tiene la misma UID que el script que está siendo ejecutado.
dl()	Esta función no está habilitada en safe-mode (modo-seguro)
backtick operator	Esta función no está habilitada en safe-mode (modo-seguro)
shell_exec() (ñalencia funcional de backticks)	Esta función no está habilitada en safe-mode (modo-seguro)
exec()	Ud. puede correr sólo ejecutables dentro delsafe_mode_exec_dir. Por razones prácticas, no está actualmente permitido tener componentes .. en la ruta del fichero ejecutable.
system()	Ud. puede correr sólo ejecutatables dentro delsafe_mode_exec_dir. Por razones prácticas, no está actualmente permitido tener componentes .. en la ruta del fichero ejecutable.
passthru()	Ud. puede correr sólo ejecutatables dentro delsafe_mode_exec_dir. Por razones prácticas, no está actualmente permitido tener componentes .. en la ruta del fichero ejecutable.
popen()	Ud. puede correr sólo ejecutatables dentro delsafe_mode_exec_dir. Por razones prácticas, no está actualmente permitido tener componentes .. en la ruta del fichero ejecutable.
mkdir()	Comprueba si el directorio que va a utilizar, tiene la misma UID que el script que está siendo ejecutado.
rmdir()	Comprueba que los archivos/directorios que va a utilizar, tengan la misma UID que el script que está siendo ejecutado.

Función	Limitaciones
rename()	Comprueba que los archivos/directorios que va a utilizar, tengan la misma UID que el script que está siendo ejecutado. Comprueba si el directorio que va a utilizar, tiene la misma UID que el script que está siendo ejecutado.
unlink()	Comprueba que los archivos/directorios que va a utilizar, tengan la misma UID que el script que está siendo ejecutado. Comprueba si el directorio que va a utilizar, tiene la misma UID que el script que está siendo ejecutado.
copy()	Comprueba que los archivos/directorios que va a utilizar, tengan la misma UID que el script que está siendo ejecutado. Comprueba si el directorio que va a utilizar, tiene la misma UID que el script que está siendo ejecutado. (en *source* y *target*)
chgrp()	Comprueba que los archivos/directorios que va a utilizar, tengan la misma UID que el script que está siendo ejecutado.
chown()	Comprueba que los archivos/directorios que va a utilizar, tengan la misma UID que el script que está siendo ejecutado.
chmod()	Comprueba que los archivos/directorios que va a utilizar, tengan la misma UID que el script que está siendo ejecutado. Además, Ud. no puede setear los bits de SUID, SGID y sticky
touch()	Comprueba que los archivos/directorios que va a utilizar, tengan la misma UID que el script que está siendo ejecutado. Comprueba si el directorio que va a utilizar, tiene la misma UID que el script que está siendo ejecutado.
symlink()	Comprueba que los archivos/directorios que va a utilizar, tengan la misma UID que el script que está siendo ejecutado. Comprueba si el directorio que va a utilizar, tiene la misma UID que el script que está siendo ejecutado. (Nota: sólo el target es comprobado)
link()	Comprueba que los archivos/directorios que va a utilizar, tengan la misma UID que el script que está siendo ejecutado. Comprueba si el directorio que va a utilizar, tiene la misma UID que el script que está siendo ejecutado. (Nota: sólo the target es comprobado)
getallheaders()	En Modo Seguro, las cabeceras que empiezan con 'authorization' (insensitivo al tipo de letra) no serán retornadas. Advertencia: esto está roto por la implementación de aol-server de **getallheaders()**!
header()	En Modo Seguro, el UID del script está agregado a la parte *realm* de la cabecera *WWW-Authenticate* si Ud. setea esta cabecera (usado por HTTP Authentication).
highlight_file(), **show_source()**	Comprueba que los archivos/directorios que va a utilizar, tengan la misma UID que el script que está siendo ejecutado. Comprueba si el directorio que va a utilizar, tiene la misma UID que el script que está siendo ejecutado. (Nota: sólo afectado desde PHP 4.2.1)
parse_ini_file()	Comprueba que los archivos/directorios que va a utilizar, tengan la misma UID que el script que está siendo ejecutado. Comprueba si el directorio que va a utilizar, tiene la misma UID que el script que está siendo ejecutado. (Nota: sólo afectado desde PHP 4.2.1)
Cualquier función que usa `php4/main/fopen_wrappers.c`	??

Capítulo 43. Usando PHP desde la línea de comando

Desde la versión 4.3.0, *PHP* soporta un nuevo tipo de *SAPI* (Interfaz De Programación De Uso Del Servidor) llamada *CLI* que significa literalmente *interfaz de línea de comando (Command Line Interface)*. Como el nombre implica, este tipo de *SAPI* se foca en la creación de aplicaciones que pueden correr desde la línea de comando (o desde el desktop también) con *PHP*. Hay algunas diferencias dentro del *CLI SAPI* y otros *SAPI* que son explicadas en este capítulo. Es importante mencionar que *CLI* y *CGI* son diferentes clases de SAPI y comparten algunas características.

La interfaz llamada *CLI SAPI* fue introducida con *PHP 4.2.0*, pero es todavía en estado experimental y tiene que ser activada explícitamente con *--enable-cli* cuando usando *./configure*. Desde *PHP 4.3.0* la interfaz *CLI SAPI* es activada automáticamente. Tu puedes usar *--disable-cli* para de-activarla.

Desde *PHP 4.3.0*, el nombre, locación, y existencia de los binarios CLI/CGI serán diferentes dependiendo en como Instales *PHP* en tu sistema. Cuando ejecutes *make*, CGI, y CLI son compilados automáticamente, y puestas como *sapi/cgi/php* y *sapi/cli/php* respectivamente, en el directorio "source" de *PHP*. Debes notar, que los dos son llamados *php*. Lo que ocurre durante el proceso *make* depende en tu línea de configuración (./configure). Si el modulo *SAPI* es seleccionado durante tu configuración, como por ejemplo *apxs*, o la opción *--disable-cgi* es usada, el *CLI* es copiado a *{PREFIX}/bin/php* durante la ejecución del comando *make install* de otras maneras el CGI es instalado aquí. Por ejemplo, si pones *--with-apxs* en tu configuración, entonces el *CLI* es copiado a *{PREFIX}/bin/php* durante *make install*. Si tu quieres sobrescribir la instalación del *CGI* binario, utiliza *make install-cli* después de usar *make install*. Alternativamente puedes especificar *--disable-cgi* en tu línea de configuración.

> **Nota:** Por que ambos *--enable-cli* y *--enable-cgi* son activados automáticamente, simplemente teniendo *--enable-cli* en tu línea de configuración no necesariamente significa que *CLI* son copiados a *{PREFIX}/bin/php* durante *make install*.

Los archivos de *PHP 4.2.0* y *PHP 4.2.3* distribuían el CLI como `php-cli.exe`, y los mantenía en el mismo directorio que el CGI `php.exe`. Empezando con *PHP 4.3.0* el archivo para windows distribuye el CLI como `php.exe` en un directorio llamado cli; o sea *cli/php.exe*.

> **Que versión de SAPI tengo?:** Desde tu línea de comando, ejecutando *php -v* te dejara saber si *php* es *CGI* o *CLI*.

Diferencias remarcables del *CLI SAPI* comparadas con otros *SAPI*s: *SAPI*s:

- En esta clase de *CGI SAPI* no hay cabeceras ("headers") escritas en el resultado ("output").

  Aunque el *CGI SAPI* provee una manera de suprimir HTTP cabeceras ("headers"), no existe una opción ñalente que los activa en el *CLI SAPI*.

  CLI automáticamente empieza en modo silencioso, la opción *-q* existe por compatibilidad con antiguos programas *CGI*.

  No cambia el directorio corriente, a ese en el cual el programa vive. La opción *-C* es mantenida por compatibilidad.

Errores son reportados en texto, no en el formato HTML.

- Hay ciertas directivas en el `php.ini` que son sobrescrita por el *CLI SAPI* por que estas no hacen mucho sentido en situaciones donde la línea de comando es usada:

Tabla 43-1. Directivas sobrescrita en `php.ini`

Directivas (directives)	*CLI SAPI* evaluó automático (default value)	Commentario (comment)
html_errors	**FALSE**	Cuando los resultados incorrectos aparecen en tu línea de comando, puede ser difícil hacer sentido de ellos con todas esas *HTML* tags, por esta razón, esta directiva es automáticamente **FALSE**.
implicit_flush	**TRUE**	Es deseoso que los resultados de **print (imprimir)()**, **echo (ecco)()** y otras relacionadas, sean inmediatamente escritas como resultados y no mantenidas en ningún buffer. Tu todavía puedes usar output buffering si tu quieres manipular los resultados proveidos automáticamente.
max_execution_time	0 (unlimited)	Debido un numero ilimitado de posibilidades de usar *PHP* en la línea de comando, el máximo tiempo de ejecución es ilimitado. Aunque aplicaciones escritas para el Internet, usualmente requieres una rápida ejecución, la clase de aplicación que es ejecutada desde la línea de comando, usualmente necesitan mas tiempo para ejecutar correctamente.
register_argc_argv	**TRUE**	Por que estas opciones son **TRUE** tu siempre necesitaras acceso al *argc* (el numero de argumentos pasados a la aplicación) y *argv* (el array de argumentos) en el *CLI SAPI*. Desde la versión 4.3.0 de *PHP*, las variables *$argc* y *$argv* son registradas y llenadas con los resultados apropiados cuando usando *CLI SAPI*. Antes de esta versión, la creación de estas variables es similar a como en *CGI* y *MODULE* versiones que requiere la PHP directiva register_globals estar *on (active)*. Sin importar la versión o la configuración de register_globals tu siempre puedes trabajar por medio de $_SERVER o *$HTTP_SERVER_VARS*. Por ejemplo: *$_SERVER['argv']*

Nota: Estas instrucciones no pueden ser iniciadas con valores que son diferentes a los de la configuración en `php.ini` o el archivo correspondiente. Esta es una limitación dada por que esos valores automáticos, son aplicados después de que todos los archivos conteniendo parámetros de configuración an sido ejecutados; PERO, esto valores pueden ser cambiados mientras to programa esta ejecutando (esto no hace sentido para todas las directivas, como por ejemplo register_argc_argv).

- Para facilitar trabajando en la línea de comando, las siguientes constantes son definidas:

Tabla 43-2. constantes especificas de CLI

Constant (constante)	Description (descripción)
STDIN	Una stream abierta hacia *stdin*. Esto nos salva de abrirla con `$stdin = fopen('php://stdin', 'r');`
STDOUT	Una stream abierta hacia *stdout*. Esto nos salva de abrirla con `$stdout = fopen('php://stdout', 'w');`
STDERR	Una stream abierta hacia *stderr*. Esto nos salva de abrirla con `$stderr = fopen('php://stderr', 'w');`

Dado lo anterior, tu no necesitas abrir, como por ejemplo, una stream hacia *stderr* manualmente, solamente necesitas usar la constante en vez de usar los recursos de la stream:

```
php -r 'fwrite(STDERR, "stderr\n");'
```

No necesitas cerrar estas stream explícitamente, desde que son cerradas automáticamente por *PHP* cuando tu programa termina.

- El *CLI SAPI* **no** cambia el directorio en el cual to estas corrientemente, al directorio donde el programa ejecutado vive!

  Ejemplo mostrando la diferencia al *CGI SAPI*:

  ```
  <?php
      /* Nuestra aplicaci&oacute;n llamada.php*/
      echo getcwd(), "\n";
  ?>
  ```

  Cuando usas la versión *CGI* el resultado es:

  ```
  $ pwd
  /tmp

  $ php -q otro_directorio/test.php
  /tmp/otro_directorio
  ```

  Esto claramente muestra que *PHP* cambia su directorio al usado por el programa que ejecutas.

  Usando el *CLI SAPI* resulta:

  ```
  $ pwd
  /tmp

  $ php -f otro_directorio/test.php
  /tmp
  ```

  Esto no da mas flexibilidad cuando escribiendo utilidades en la línea de comando con *PHP*.

  Nota:

Puedes obtener acceso a la lista de opciones proveida por *PHP* ejecutando *PHP* con el *-h*switch:

```
Usage: php [options] [-f] <file> [args...]
       php [options] -r <code> [args...]
       php [options] [-- args...]
  -s               Display colour syntax highlighted source.
  -w               Display source with stripped comments and whitespace.
  -f <file>        Parse <file>.
  -v               Version number
  -c <path>|<file> Look for php.ini file in this directory
  -a               Run interactively
  -d foo[=bar]     Define INI entry foo with value 'bar'
  -e               Generate extended information for debugger/profiler
  -z <file>        Load Zend extension <file>.
  -l               Syntax check only (lint)
  -m               Show compiled in modules
  -i               PHP information
  -r <code>        Run PHP <code> without using script tags <?..?>
  -h               This help

  args...          Arguments passed to script. Use -- args when first argument
                   starts with - or script is read from stdin
```

Aunque los resultados anteriores siempre serán dados en ingles, a continuación te daré una lista que probablemente sera muy útil:

```
Usage: php [options] [-f] <file> [args...]
         php [options] -r <code> [args...]
         php [options] [-- args...]
    -s               Display colour syntax highlighted source.
                     (colorear el sintaxis en el c&oacute;digo)
    -w               Display source with stripped comments and whitespace.
                     (remueve los comentarios y espacios del c&oacute;digo)
    -f <file>        Parse <file>.
                     (analiza <file>)
    -v               Version number
                     (la versi&oacute;n de PHP que estas usando)
    -c <path>|<file> Look for php.ini file in this directory
                     (usa el php.ini archivo localizado aqu&iacute;)
    -a               Run interactively
                     (interactivo)
    -d foo[=bar]     Define INI entry foo with value 'bar'
                     (define foo con el valor 'bar' en php.ini)
    -e               Generate extended information for debugger/profiler
                     (genera mas informaci&oacute;n para el debugger/profiler)
    -z <file>        Load Zend extension <file>.
                     (inicia las exenciones Zend <archive>)
    -l               Syntax check only (lint)
                     (Mira al sintaxis (lint))
    -m               Show compiled in modules
                     (muestra los m&oacute;dulos compilados)
    -i               PHP information
                     (informaci&oacute;n PHP)
    -r <code>        Run PHP <code> without using script tags <?..?>
                     (ejecuta PHP <code> sin usar las tags <?..?> en el script
    -h               This help
                     (estas opciones)
    args...          Arguments passed to script. Use -- args when first argument
                     starts with - or script is read from stdin
                     (Argumentos pasados al programa. Usa -- args cuando el primer
                     argumento empieza con - o tu programa es le&iacute;do directamente
```

El *CLI SAPI* tiene tres diferentes maneras de obtener el código *PHP* que tu quieres ejecutar:

1. Puedes decir a *PHP* que ejecute ciertos archivos.

   ```
   php my_script.php

   php -f my_script.php
   ```

   En estos dos ejemplos (aunque utilices el switch *-f* o no) ejecutan el archivo `my_script.php`. Tu puedes escoger cualquier archivo para ejecutar - tus programas *PHP* no tienen que terminar

con la exención `.php` y pueden tener cualquier otra exención tu desees.

2. Pasa el código *PHP* para que sea ejecutado directamente en la línea de comando.

   ```
   php -r 'print_r(get_defined_constants());'
   ```
   Debes tener cuidado en reguardo a las substituciones variables en tu línea de comando y usando comillas(").

   > **Nota:** Deves ponerle atención al ejemplo, y notaras que no tiene tags ni al principio ni al final, el comando *-r* simplemente no las usa. Usando las tags te dará un error cuando trates de ejecutar el programa.

3. Provee el código *PHP* para ejecutar por medio de *stdin*

   Esto te da la habilidad de dinámicamente crear código *PHP* y mandarlo al programa, como por ejemplo a continuación:
   ```
   $ some_application | some_filter | php | sort -u >final_output.txt
   ```
Tu no puedes combinar ninguna de las tres formas de ejecutar el código.

Como cualquier aplicación ejecutada en la línea de comando, el *PHP* binario acepta un numero de argumentos y tu programa también puede recibir argumentos. El numero de argumentos que pueden ser pasados a tu programa no es limitado por *PHP* (la línea de comando tiene limitaciones en el numero de símbolos que pueden ser pasados; usualmente tu nunca alcanzarías este limite). Los argumentos pasados a tu programa, están disponibles en tu array global *$argv*. El índex cero ("0") siempre contiene el nombre de tu programa (que es - en caso de código que esta viniendo por medio del input estándar, o del switch en la línea de comando *-r*. La segunda variable global registrada es *$argc* y contiene el numero de elementos en el array *$argv* (*no* el numero de argumentos pasados as programa).

Mientras el argumento que tu quieres pasas a tu programa no comienza con -, no tienes que esperar por nada especial. Pero si el argumento empieza con -, te puede generar problemas, por que *PHP* pensara que tiene que procesarlo. Para prevenir esto, usa la lista separadora de argumentos: --. Después de que el separador a sido procesado, cada siguiente argumento es pasado sin tocar a tu programa.

```
# This will not execute the given code but will show the PHP usage
# Esto no ejecutara el c&oacute;digo pero PHP mostrara el uso
$ php -r 'var_dump($argv);' -h
Usage: php [options] [-f] <file> [args...]
[...]

# This will pass the '-h' argument to your script and prevent PHP from showing it's usa
# passaremos el argumento '-h' a tu programa y prevenira que PHP demuestre su uso
$ php -r 'var_dump($argv);' -- -h
array(2) {
  [0]=>
  string(1) "-"
  [1]=>
  string(2) "-h"
}
```
Pero, existe otra manera de usar *PHP* en la línea de comando. Tu puedes escribir un programa donde la primera línea empieza con *#!/usr/bin/php (donde /usr/bin/php es la locación de php)*. Después de esto, tu puedes usar *PHP* común y corriente. Una vez que tu le as dado permiso de ejecución a tu programa (por ejemplo *+x test*) tu programa puede ser ejecutado como si fuera digamos un programa en perl:

```
#!/usr/bin/php
<?php
    var_dump($argv);
?>
```

Asumiremos que el archivo es llamado *test*, y esta en el mismo directorio en el cual to estas, en ese caso, podemos hacer lo siguiente.

```
$ chmod 755 test
$ ./test -h -- foo
array(4) {
  [0]=>
  string(6) "./test"
  [1]=>
  string(2) "-h"
  [2]=>
  string(2) "--"
  [3]=>
  string(3) "foo"
}
```

Como puedes ver, en este caso no atención es dada a pasar los parámetros que comiensen con - en tu programa.

Tabla 43-3. Opciones en la línea de comando

Opciones	Descripcion
-s	colora el sintaxis de tu código Esta opción usa un mecanismo interno para ejecutar el archivo, y produce una versión coloreada e normal. Nota que todo lo que hace es generar un bloque de *<code> [...] </code> HTML* tags, no **Nota:** Esta opción no trabaja en conjunto con *-r*.
-w	Te mostrara tu código sin comentarios ni espacios blancos. **Nota:** Esta opción no trabaja en conjunto con *-r*.
-f	Ejecuta el archivo indicado en la opción *-f*. Esta opción es opcional y puede ser excluida. Solamer necesita ser ejecutado es suficiente.

-v	Escribe la version de PHP, PHP SAPI y Zend al output normal, por ejemplo:
	`$ php -v`
	`PHP 4.3.0 (cli), Copyright (c) 1997-2002 The PHP Group`
	`Zend Engine v1.3.0, Copyright (c) 1998-2002 Zend Technologies`

-c	Con esta opción uno puede especificar el directorio donde encontraremos el `php.ini` archivo, o única del mismo (la cual no tiene que ser llamada `php.ini`), por ejemplo:
	`$ php -c /costumatisado/directorio/ mi_programa.php`
	
	`$ php -c /customatisado/directorio/customatisado-archivo.ini mi_programa.php`

-a	Corre PHP interactivamente.

Opciones	Descripcion

-d	Esta opción te hayudara a crear el valor de cualquier directiva de configuración permitidas en el p
	`-d directiva__de_configuracion [=valor]`
	Ejemplos:
	`# Omitting the value part will set the given configuration directive to "1"`
	`# Omitiendo la parte relacionada al valor, le asignara a la directiva de confi`
	`$ php -d max_execution_time -r '$foo = ini_get("max_execution_time"); var_dump`
	`string(1) "1"`
	
	`# Passing an empty value part will set the configuration directive to ""`
	`# Pasando un valor vac&iacute;o, le asignara a la directiva de configuraci&oac`
	`php -d max_execution_time= -r '$foo = ini_get("max_execution_time"); var_dump`
	`string(0) ""`
	
	`# The configuration directive will be set to anything passed after the '=' cha`
	`# la directiva de configuraci&oacute;n sera asignada a todo pasada el "=" simb`
	`$  php -d max_execution_time=20 -r '$foo = ini_get("max_execution_time"); var_`
	`string(2) "20"`
	`$  php -d max_execution_time=doesntmakesense -r '$foo = ini_get("max_executior`
	`string(15) "doesntmakesense"`

-e	Generando mas información para el debugger/profiler.
-z	Activa las extensiones Zend. Si solamente un archivo es dado, *PHP* tratará de activar estas extens directorio predeterminado donde esté la biblioteca en su sistema (Usualmente especificado `/etc` Pasando el nombre del archivo con descripción absoluta de la ubicación de sus archivos, no usará archivo conteniendo la información de estos directorios, le dira a *PHP* que solamente trate de acti directorio donde te encuentras
-l	Esta opción proveerá una forma conveniente para marcar tu sintaxis en tu código. En caso de suce *detected in <filename>* (no errores de sintaxis fueron detectados) es escrito en tu output normal, y código *0*. En caso de problemas, el texto *Errors parsing <filename>*, en adición al la forma intern son escritos como output normal y tu línea de comando recibirá el código *255* Esta opción no encontrara errores fatales (como por ejemplo funciones indefinida), usa *-f* si tu qui errores también. **Nota:** Esta opción no trabaja en conjunción con *-r*

-m	Usando esta opción, *PHP* imprime sus módulos internos (y activados) usados por PHP y Zend:
	`$ php -m`
	`[PHP Modules]`
	`xml`
	`tokenizer`
	`standard`
	`session`
	`posix`
	`pcre`
	`overload`
	`mysql`
	`mbstring`
	`ctype`
	
	`[Zend Modules]`

-i	Esta opción llama *phpinfo*, e imprime los resultado. Si *PHP* no esta trabajando correctamente, es opción observes si algún mensaje es imprimido antes de, o en medio de la información dada por e importante que entiendas que el mensaje imprimido es en *HTML* y por esta razón grandecito.

<table>
<tr><th>Opciones</th><th>Descripcion</th></tr>
<tr><td>-r</td><td>
Esta opción te ayudara a ejecutar <i>PHP</i> directamente desde la línea de comando. Las etiquetas que de tu programa (<i><?php</i> y <i>?></i>) no son necesarias y causaran errores si las pones en tu código.

Nota: Debes tener cuidado cuando usando esta forma de PHP para que no crees conflicto c variables usada por la línea de comando.

Ejemplos de errores
<pre>$ php -r "$foo = get_defined_constants();"
Command line code(1) : Parse error - parse error, unexpected '='</pre>
El problema aquí es que sh/bash esta haciendo una substitución de variables, aunque las comillas variable $foo probablemente no esta definida, esta no se inflara en ninguna dirección, el resultado para que ejecute realmente lee:
<pre>$ php -r " = get_defined_constants();"</pre>
La forma correcta de hacer esto, seria usando solamente una comilla (′), variables usando solamer por sh/bash.
<pre>$ php -r '$foo = get_defined_constants(); var_dump($foo);'
array(370) {
  ["E_ERROR"]=>
  int(1)
  ["E_WARNING"]=>
  int(2)
  ["E_PARSE"]=>
  int(4)
  ["E_NOTICE"]=>
  int(8)
  ["E_CORE_ERROR"]=>
  [...]</pre>
Si tu no estas usando sh/bash, también puedes encontrar otros problemas. Por favor reporta estos phpdocs@lists.php.net Tu también puedes tener problemas si tratas de poner variables en tu códig símbolos de escape. Te hemos advertido

Nota: -r esta listo en CLI SAPI y no en el CGI SAPI.
</td></tr>
<tr><td>-h</td><td>Con esta opción, tu puedes obtener información acerca de las opciones describías anteriormente, sus funciones.</td></tr>
</table>

PHP puede ejecutar tus programas absolutamente independiente de tu servidor de páginas de web. Si tu usas Unix, tu puedes añadir una línea especial al principio de tu programa, y hacerlo ejecutable, para que el sistema sepa que programa debe ejecutar tu nueva creación. Si usas windows, tu puedes asociar tu programa con *php.exe* para que solamente tengas que ejecutarlo como harías con otros programas bajo windows, también puedes crear un "batch" archivo para ejecutar tu programa por medio de *PHP*. La primera línea que usaste para hacer que tu programa funcione en Unix, no le ara daño a tu programa cuando ejecutad bajo windows, pero de esta manera puedes crear programas que puedes ser usados bajo las dos plataformas. A continuación te daremos un ejemplo:

Ejemplo 43-1. Programa para correr en la línea do comando (script.php)

```
#!/usr/bin/php
<?php

if ($argc != 2 || in_array($argv[1], array('--help', '-help', '-h', '-?'))) {
?>

Este es un programa en php entendido para la l&iacute;nea de
comando con una opci&oacute;n.

  Usage:
  <?php echo $argv[0]; ?> <option>

  <option> puede ser cualquier palabra que tu quieras
  imprimir. Con la opci&oacute;n --help, -help -h or -?, tu puedes
  obtener esta informaci&oacute;n

<?php
} else {
    echo $argv[1];
}
?>
```

En el programa anterior, usamos una línea especial como nuestra primera línea, para indicar que archivo deber ser ejecutado por PHP. Nosotros trabajamos con una versión de CLI aquí, por eso, no tendremos cabeceras de HTTP imprimidas. Hay dos variables que puedes usar cuando escribiendo aplicaciones en la línea de comando en PHP: *$argc* y *$argv*. La primera es el numero de argumentos mas uso (el nombre del programa siendo ejecutado). La segunda es un array conteniendo los argumentos, empezando con el programa nombre, y el numero cero "0" (*$argv[0]*).

En el programa anterior chequeamos si habían mas, o menos de dos argumentos. También trata de ver si *--help*, *-help*, *-h* o *-?*, son llamados, e imprime el mensaje de ayuda.

Si tu quieres ejecutar el programa anterior en Unix, tu tienes que hacerlo ejecutable, y simplemente llamado *script.php echo this* o *script.php -h*. En windows, tu puedes hacer un batch archivo para alcanzar estos resultados:

Ejemplo 43-2. Archivo batch para ejecutar el programa php (script.bat)

```
@c:\php\cli\php.exe script.php %1 %2 %3 %4
```

Asumiendo que llamaste el programa descrito anteriormente `script.php` , Y que tienes tu CLI `php.exe` en `c:\php\cli\php.ese` este archivo batch, lo ejecutara para ti con las funciones añadidas: *script.bat echo this* o *script.bat -h*.

Mira también la documentación de Readline para mas funciones que puedes usar para incrementar tus opciones en este sujeto.

VI. Referencia de funciones

Tabla de contenidos

I. Funciones específicas de Apache

Introducción

Estas funciones están disponibles solamente cuando PHP se ejecuta como módulo de Apache 1.x.

Instalación

Información sobre la instalación de PHP con Apache se puede encontrar en el capítulo sobre instalación en la sección sobre Apache

---

Configuración en tiempo de ejecución

El comportamiento del módulo PHP de Apache está sujeto a los parámetros ajustados en `php.ini`. Los parámetros ajustados mediante php_flag en el archivo de configuración del servidor o archivos `.htaccess` locales, tendrán preferencia sobre aquellos ajustados en `php.ini`.

Ejemplo 1. Desactivar el intérprete PHP en un directorio utilizando `.htaccess`

```
php_flag engine off
```

Tabla 1. Opciones de configuración de Apache

Nombre	Por defecto	Modificable	Función
engine	On	PHP_INI_ALL	habilita o desactiva el intérprete PHP
child_terminate	Off	PHP_INI_ALL	especifica si los scripts PHP pueden requerir la terminación del proceso hijo al acabar un requerimiento. Véase también **apache_child_terminate()**
last_modified	Off	PHP_INI_ALL	enviar la fecha de modificación de los scripts PHP como la fecha de la última modifición en la cabecera del requerimiento actual
xbithack	Off	PHP_INI_ALL	interpretar los archivos cuyo bit ejecutable esté fijado a PHP, independientemente de su extensión

A continuación se presenta una corta explicación de las directivas de configuración.

engine **boolean**

Esta directiva realmente sólo es útil cuando PHP es un módulo de Apache. Se utiliza para sitios que quieran activar o desactivar el intérprete de PHP en función del directorio o del host-virtual. Añadiendo `engine off` en los lugares apropiados del archivo `httpd.conf`, PHP puede ser habilitado o desactivado.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Esta extensión no tiene ninguna constante definida.

Tabla de contenidos

apache_child_terminate

(PHP 4 >= 4.0.5, PHP 5)

apache_child_terminate -- Terminar un proceso de apache una vez concluido el requerimiento en ejecució

Descripcíon

bool **apache_child_terminate** (void)

apache_child_terminate() registrará el proceso Apache que esté ejecutando el requeriminto PHP actual para su terminacióon una vez que la ejecución del código haya finalizado. Puede ser utilizado para terminar un proceso una vez que un script con un alto consumo de memoria haya sido ejecutado, dado que la memoria únicamente sera liberada de forma interna, pero no será devuelta al sistema operativo.

> **Nota:** La disponibilidad de esta característica se controla mediante la directiva *child_terminate* `php.ini`, que por defecto está establecida como *off* (desactivada).
>
> Esta característica tampoco está disponible en las versiones "multi-thread" de Apache, como por ejemplo la versión win32.

Véase también la función **exit()**.

apache_get_modules

(PHP 4 >= 4.3.2, PHP 5)

apache_get_modules -- Obtiene una lista de los modulos cargados con el servidor Apache

Descripción

matrix **apache_get_modules** (void)

Esta función regresa una matriz, con los modulos que han sido cargados por Apache.

Ejemplo 1. Ejemplo de apache_get_modules()

```
<?php
print_r(apache_get_modules());
?>
```

El ejemplo anterior ejemplo, producirá algo similar a:

```
Array
(
    [0] => core
    [1] => http_core
    [2] => mod_so
    [3] => sapi_apache2
    [4] => mod_mime
    [5] => mod_rewrite
)
```

apache_get_version

(PHP 4 >= 4.3.2, PHP 5)

apache_get_version -- Obtiene la version de Apache

Descripción

cadena **apache_get_version** (void)

apache_get_version() regresa la version de Apache como una cadena, o **FALSE** en caso de que falle.

Ejemplo 1. Ejemplo de apache_get_version()

```
<?php
$version = apache_get_version();
echo "$version \n <br />";
?>
```

El ejemplo anterior producirá algo similar a:

```
Apache/1.3.29 (Unix) PHP/4.3.4
```

Vea también **phpinfo()**.

apache_getenv

(PHP 4 >= 4.3.0, PHP 5)

apache_getenv -- Obtiene una variable subprocess_env de Apache

Descripción

cadena **apache_getenv** (cadena variable [, bool walk_to_top])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

apache_lookup_uri

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

apache_lookup_uri -- Realiza una petición parcial por la URI especificada y devuelve toda la información sobre ella

Descripción

object **apache_lookup_uri** (string nombre_archivo)

Esta función realiza una petición parcial por una URI. Tan solo llega a obtener toda la información importante sobre el recurso dado y devuelve esta información en una clase. Las propiedades de la

clase devuelta son:

status
the_request
status_line
method
content_type
handler
uri
filename
path_info
args
boundary
no_cache
no_local_copy
allowed
send_bodyct
bytes_sent
byterange
clength
unparsed_uri
mtime
request_time

Ejemplo 1. Ejemplo de apache_lookup_uri()

```
<?php
$info = apache_lookup_uri('index.php?var=valor');
print_r($info);

if (file_exists($info->filename)) {
    echo '&iexcl;el archivo existe!';
}
?>
```

El ejemplo anterior producirá una salida similar a:

```
stdClass Object
(
    [status] => 200
    [the_request] => GET /dir/archivo.php HTTP/1.1
    [method] => GET
    [mtime] => 0
    [clength] => 0
    [chunked] => 0
    [content_type] => application/x-httpd-php
    [no_cache] => 0
    [no_local_copy] => 1
    [unparsed_uri] => /dir/index.php?var=valor
    [uri] => /dir/index.php
    [filename] => /home/htdocs/dir/index.php
    [args] => var=valor
    [allowed] => 0
    [sent_bodyct] => 0
    [bytes_sent] => 0
    [request_time] => 1074282764
)
&iexcl;el archivo existe!
```

Nota: apache_lookup_uri() sólo funciona cuando PHP se encuentra instalado como un módulo de Apache.

apache_note

(PHP 3>= 3.0.2, PHP 4 , PHP 5)

apache_note -- Obtener y establecer las notas de petición de apache

Descripción

string **apache_note** (string nombre_nota [, string valor_nota])

apache_note() es una función específica de Apache que obtiene y define valores en la tabla *notes* de una petición. Si es llamada con un argumento, devuelve el valor actual de la nota *nombre_nota*. Si es llamada con dos argumentos, define el valor de la nota *nombre_nota* a *valor_nota* y devuelve el valor previo de la nota *nombre_nota*.

apache_request_headers

(PHP 4 >= 4.3.0, PHP 5)

apache_request_headers -- Obtener todas las cabeceras HTTP

Descripción

array **apache_request_headers** (void)

apache_request_headers() devuelve una matriz asociativa de todas las cabeceras HTTP en la petición actual. Esta función está disponible únicamente cuando PHP se ejecuta como un módulo de Apache.

Ejemplo 1. Ejemplo de apache_request_headers()

```
<?php
$headers = apache_request_headers();

foreach ($headers as $header => $value) {
    echo "$header: $value <br />\n";
}
?>
```

> **Nota:** Con anterioridad a PHP 4.3.0, **apache_request_headers()** se denominaba **getallheaders()**. A partir de PHP 4.3.0, **getallheaders()** es un alias para **apache_request_headers()**.
>
> **Nota:** También se puede obtener el valor de las variables CGI comunes a partir de las variables de entorno, lo cual funciona independientemente de que se esté utilizando PHP como un módulo de Apache. Utiliza **phpinfo()** para ver una lista de todas las variables de entorno disponibles.

apache_reset_timeout

(no version information, might be only in CVS)

apache_reset_timeout -- Reset the Apache write timer

Description

bool **apache_reset_timeout** (void)

apache_reset_timeout() resets the Apache write timer, which defaults to 300 seconds. With *set_time_limit(0); ignore_user_abort(true)* and periodic **apache_reset_timeout()** calls, Apache can theoretically run forever.

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

> **Nota:** This functions is just available for Apache 1.

> **Nota:** Esta función no está habilitada en safe-mode (modo-seguro)

apache_response_headers

(PHP 4 >= 4.3.0, PHP 5)

apache_response_headers -- Obtener todas las cabeceras HTTP de respuesta

Descripción

array **apache_response_headers** (void)

Devuelve una matriz con todas las cabeceras de respuesta de Apache. Esta característica sólo está disponible a partir de PHP 4.3.0.

Véase también **getallheaders()** y **headers_sent()**.

apache_setenv

(PHP 4 >= 4.2.0, PHP 5)

apache_setenv -- fijar una variable subprocess_env de Apache

Description

int **apache_setenv** (string variable, string value [, bool walk_to_top])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ascii2ebcdic

(PHP 3>= 3.0.17)

ascii2ebcdic -- Traducir una cadena en ASCII a EBCDIC

Descripción

int **ascii2ebcdic** (string ascii_str)

ascii2ebcdic() es una función específica de Apache que únicamente está disponible en sistemas operativos basados en EBCDIC (OS/390, BS2000). Traduce la cadena *ascii_str* codificada en ASCII a su representació ñalente EBCDIC (a prueba de binario), y devuelve el resultado.

Véase también la función inversa **ebcdic2ascii()**

ebcdic2ascii

(PHP 3>= 3.0.17)

ebcdic2ascii -- Traduce una cadena en EBCDIC a ASCII

Descripción

int **ebcdic2ascii** (string ebcdic_str)

ebcdic2ascii() es una función específica de Apache que únicamente está disponible en sistemas operativos basados en EBCDIC (OS/390, BS2000). Traduce la cadena *ebcdic_str* codificada en EBCDIC a su representació ñalente ASCII (a prueba de binario), y devuelve el resultado.

Véase también la función inversa **ascii2ebcdic()**

getallheaders

(PHP 3, PHP 4 , PHP 5)

getallheaders -- Recuperar todas las cabeceras de petición HTTP

Descripción

array **getallheaders** (void)

getallheaders() es un alias para **apache_request_headers()**. Devolverá una matriz asociativa con todas las cabeceras HTTP en la petición actual. Por favor lea la documentación de **apache_request_headers()** para más información sobre cómo trabaja esta función.

> **Nota:** En PHP 4.3.0, **getallheaders()** se hizo un alias de **apache_request_headers()**. Básicamente, fue renombrada. Esto debido a que esta función sólo trabaja cuando PHP es compilado como un módulo de Apache.
>
> **Nota:** A partir de PHP 4.3.3 se puede usar tambien esta funcion con el modulo de servidor NSAPI de los servidores web, Netscape/iPlanet/SunONE.

Vea también **apache_request_headers()**.

virtual

(PHP 3, PHP 4 , PHP 5)

virtual -- Realizar una sub-petición de Apache

Descripción

int **virtual** (string nombre_archivo)

virtual() es una función específica de Apache que es ñalente a <!--#include virtual...--> en mod_include. Realiza una sub-petición de Apache. Es útil para incluir scripts CGI o archivos .shtml, o cualquier otra cosa que quisiera procesar a través de Apache. Note que para un script CGI, el script debe generar cabeceras CGI válidas. Esto quiere decir que, por lo menos, debe generar una cabecera Content-type.

Para ejecutar la sub-petición, todos los búferes son terminados y volcados al navegador, y las cabeceras pendientes son enviadas también.

Aviso
Esta función sólo trabaja cuando PHP se compila como un módulo de Apache, ya que usa la API de Apache para realizar las sub-peticiones. La cadena de query puede ser pasada al archivo incluido pero la variable *$_GET* es copiada desde el script padre, y solo *$_SERVER ['QUERY_STRING']* se llena con la cadena de query pasada. Puede que la cadena query sólo pueda ser pasada cuando se usa Apache 2. El archivo solicitado no será listado en el archivo de registro (log) \access de Apache.

A partir de PHP 4.0.6, puede usar **virtual()** sobre archivos PHP. Sin embargo, típicamente es mejor usar **include()** o **require()** si desea incluier otro archivo PHP.

> **Nota:** A partir de PHP 4.3.3 se puede usar tambien esta funcion con el modulo de servidor NSAPI de los servidores web, Netscape/iPlanet/SunONE.

II. Advanced PHP debugger

Introducción

APD is the Advanced PHP Debugger. It was written to provide profiling and debugging capabilities for PHP code, as well as to provide the ability to print out a full stack backtrace. APD supports interactive debugging, but by default it writes data to trace files. It also offers event based logging so that varying levels of information (including function calls, arguments passed, timings, etc.) can be turned on or off for individual scripts.

Atención
APD is a Zend Extension, modifying the way the internals of PHP handle function calls, and thus may or may not be compatible with other Zend Extensions (for example Zend Optimizer).

Instalación

APD is currently available as a PECL extension from http://pecl.php.net/package/apd. Make sure you have installed the CGI version of PHP and it is available in your current path along with the phpize script.

Run the following command to download, build, and install the latest stable version of APD:

```
pear install apd
```

This automatically installs the APD Zend module into your PHP extensions directory. It is not mandatory to keep it there; you can store the module in any directory PHP can read as long as you set the zend_extension parameter accordingly.

Windows users can download the extension dll `php_apd.dll` from http://snaps.php.net/win32/PECL_STABLE/.

In your INI file, add the following lines:

```
zend_extension = /absolute/path/to/apd.so
apd.dumpdir = /absolute/path/to/trace/directory
apd.statement_trace = 0
```

Depending on your PHP build, the zend_extension directive can be one of the following:

```
zend_extension              (non ZTS, non debug build)
zend_extension_ts           (    ZTS, non debug build)
zend_extension_debug        (non ZTS,     debug build)
zend_extension_debug_ts     (    ZTS,     debug build)
```

Building on Win32

To build APD under Windows you need a working PHP compilation environment as described on http://php.net/ -- basically, it requires you to have Microsoft Visual C++, win32build.zip, bison/flex, and some know how to get it to work. Also ensure that adp.dsp has DOS line endings; if it has unix line endings, Microsoft Visual C++ will complain about it.

Configuración en tiempo de ejecución

A continuación se presenta una corta explicación de las directivas de configuración.

apd.dumpdir **string**

> Sets the directory in which APD writes profile dump files. You can specify an absolute path or a relative path.
>
> You can specify a different directory as an argument to **apd_set_pprof_trace()**.

apd.statement_trace **boolean**

> Specfies whether or not to do per-line tracings. Turning this on (1) will impact the performance of your application.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Esta extensión no tiene ninguna constante definida.

How to use PHP-APD in your scripts

1. As the first line of your PHP script, call the apd_set_pprof_trace() function to start the trace:

   ```
   apd_set_pprof_trace();
   ```

   You can insert the line anywhere in your script, but if you do not start tracing at the beginning of your script you discard profile data that might otherwise lead you to a performance bottleneck.

2. Now run your script. The dump output will be written to `apd.dumpdir/pprof_pid.ext`.

   > **Sugerencia:** If you're running the CGI version of PHP, you will need to add the '-e' flag to enable extended information for apd to work properly. For example: `php -e -f script.php`

3. To display formatted profile data, issue the **pprofp** command with the sort and display options of your choice. The formatted output will look something like:

   ```
   bash-2.05b$ pprofp -R /tmp/pprof.22141.0

   Trace for /home/dan/testapd.php
   Total Elapsed Time = 0.00
   Total System Time  = 0.00
   Total User Time    = 0.00


   Real         User        System              secs/    cumm
   %Time (excl/cumm)  (excl/cumm)  (excl/cumm) Calls    call    s/call  Memory Usage Nam
   ---------------------------------------------------------------------------------
   100.0 0.00 0.00  0.00 0.00  0.00 0.00     1  0.0000   0.0009            0 main
   56.9 0.00 0.00  0.00 0.00  0.00 0.00     1  0.0005   0.0005            0 apd_set_pprof
   28.0 0.00 0.00  0.00 0.00  0.00 0.00    10  0.0000   0.0000            0 preg_replace
   14.3 0.00 0.00  0.00 0.00  0.00 0.00    10  0.0000   0.0000            0 str_replace
   ```

   The -R option used in this example sorts the profile table by the amount of real time the script spent executing a given function. The "cumm call" column reveals how many times each function was called, and the "s/call" column reveals how many seconds each call to the function required, on average.

4. To generate a calltree file that you can import into the KCacheGrind profile analysis application, issue the **pprof2calltree** comand.

Contact information

If you have comments, bugfixes, enhancements or want to help developing this beast, you can send

an mail to apd@mail.communityconnect.com. Any help is very welcome.

Tabla de contenidos

apd_breakpoint

(no version information, might be only in CVS)

apd_breakpoint -- Stops the interpreter and waits on a CR from the socket

Description

void **apd_breakpoint** (int debug_level)

This can be used to stop the running of your script, and await responses on the connected socket. To step the program, just send enter (a blank line), or enter a php command to be executed. A typical session using tcplisten would look like this.

```
bash#tcplisten localhost 7777

APD - Advanced PHP Debugger Trace File
-------------------------------------------------------------------------
Process Pid (6118)
Trace Begun at Sun Mar 10 23:13:12 2002
-------------------------------------------------------------------------
(  0.000000): apd_set_session_trace called at /home/alan/Projects/project2/test.
php:5
(  0.074824): apd_set_session_trace_socket() at /home/alan/Projects/project2/tes
t.php:5 returned.  Elapsed (0.074824)
(  0.074918): apd_breakpoint() /home/alan/Projects/project2/test.php:7
              ++ argv[0] $(??) = 9
apd_breakpoint() at /home/alan/Projects/project2/test.php:7 returned.  Elapsed (
-2089521468.1073275368)
>\n
statement: /home/alan/Projects/project2/test.php:8
>\n
statement: /home/alan/Projects/project2/test.php:8
>\n
statement: /home/alan/Projects/project2/test.php:10
>apd_echo($i);
EXEC: apd_echo($i);
0
>apd_echo(serialize(apd_get_active_symbols()));
EXEC:  apd_echo(serialize(apd_get_active_symbols()));
a:47:{i:0;s:4:"PWD";i:1;s:10:"COLORFGBG";i:2;s:11:"XAUTHORITY";i:3;s:14:"
COLORTERM_BCE";i:4;s:9:"WINDOWID";i:5;s:14:"ETERM_VERSION";i:6;s:16:"SE
SSION_MANAGER";i:7;s:4:"PS1";i:8;s:11:"GDMSESSION";i:9;s:5:"USER";i:10;s:5:"
MAIL";i:11;s:7:"OLDPWD";i:12;s:5:"LANG";i:13;s:10:"COLORTERM";i:14;s:8:"DISP
LAY";i:15;s:8:"LOGNAME";i:16;s:6:"
>apd_echo(system('ls /home/mydir'));
........
>apd_continue(0);
```

apd_callstack

(no version information, might be only in CVS)

apd_callstack -- Returns the current call stack as an array

Description

array **apd_callstack** (void)

Returns the current call stack as an array

Ejemplo 1. apd_callstack() example

```
<?php
print_r(apd_callstack());
?>
```

apd_clunk

(no version information, might be only in CVS)

apd_clunk -- Throw a warning and a callstack

Description

void **apd_clunk** (string warning [, string delimiter])

Behaves like perl's Carp::cluck. Throw a warning and a callstack. The default line delimiter is "
\n".

Ejemplo 1. apd_clunk() example

```
<?php
apd_clunk("Some Warning","<P>");
?>
```

apd_continue

(no version information, might be only in CVS)

apd_continue -- Restarts the interpreter

Description

void **apd_continue** (int debug_level)

Usually sent via the socket to restart the interpreter.

Ejemplo 1. apd_continue() example

```
<?php
apd_continue(0);
?>
```

apd_croak

(no version information, might be only in CVS)

apd_croak -- Throw an error, a callstack and then exit

Description

void **apd_croak** (string warning [, string delimiter])

Behaves like perl's Carp::croak. Throw an error, a callstack and then exit. The default line delimiter is "
\n".

Ejemplo 1. apd_croak() example

```
<?php
apd_croak("Some Warning","<P>");
?>
```

apd_dump_function_table

(no version information, might be only in CVS)

apd_dump_function_table -- Outputs the current function table

Description

void **apd_dump_function_table** (void)

Outputs the current function table.

Ejemplo 1. apd_dump_function_table() example

```
<?php
apd_dump_function_table();
?>
```

apd_dump_persistent_resources

(no version information, might be only in CVS)

apd_dump_persistent_resources -- Return all persistent resources as an array

Description

array **apd_dump_persistent_resources** (void)

Return all persistent resources as an array.

Ejemplo 1. apd_dump_persistent_resources() example

```
<?php
print_r(apd_dump_persistent_resources());
?>
```

apd_dump_regular_resources

(no version information, might be only in CVS)

apd_dump_regular_resources -- Return all current regular resources as an array

Description

array **apd_dump_regular_resources** (void)

Return all current regular resources as an array.

Ejemplo 1. apd_dump_regular_resources() example

```
<?php
print_r(apd_dump_regular_resources());
?>
```

apd_echo

(no version information, might be only in CVS)

apd_echo -- Echo to the debugging socket

Description

void **apd_echo** (string output)

Usually sent via the socket to request information about the running script.

Ejemplo 1. apd_echo() example

```
<?php
apd_echo($i);
?>
```

apd_get_active_symbols

(no version information, might be only in CVS)

apd_get_active_symbols -- Get an array of the current variables names in the local scope

Description

array **apd_get_active_symbols** ( )

Returns the names of all the variables defined in the active scope, (not their values)

Ejemplo 1. apd_get_active_symbols() example

```
<?php
apd_echo(apd_get_active_symbols());
?>
```

apd_set_pprof_trace

(no version information, might be only in CVS)

apd_set_pprof_trace -- Starts the session debugging

Description

void **apd_set_pprof_trace** ([string dump_directory])

Starts debugging to {dump_directory}/pprof_{process_id}, if dump_directory is not set, then the apd.dumpdir setting from the `php.ini` file is used.

Ejemplo 1. apd_set_pprof_trace() example

```
<?php
apd_set_pprof_trace();
?>
```

apd_set_session_trace

(no version information, might be only in CVS)

apd_set_session_trace -- Starts the session debugging

Description

void **apd_set_session_trace** (int debug_level [, string dump_directory])

Starts debugging to {dump_directory}/apd_dump_{process_id}, if dump_directory is not set, then the apd.dumpdir setting from the `php.ini` file is used.

debug_level is an integer which is formed by adding together the following values:

```
FUNCTION_TRACE      1
        ARGS_TRACE          2
        ASSIGNMENT_TRACE    4
        STATEMENT_TRACE     8
        MEMORY_TRACE        16
        TIMING_TRACE        32
        SUMMARY_TRACE       64
```

I would seriously not recommend using MEMORY_TRACE. It is very slow and does not appear to be accurate (great, huh?) also ASSIGNMENT_TRACE is not implemented. So, to turn on all functional traces (TIMING, FUNCTIONS, ARGS SUMMARY (like strace -c)) use the value 99

Ejemplo 1. apd_set_session_trace() example

```
<?php
apd_set_session_trace(99);
?>
```

apd_set_session

(no version information, might be only in CVS)

apd_set_session -- Changes or sets the current debugging level

Description

void **apd_set_session** (int debug_level)

This can be used to increase or decrease debugging in a different area of your application,.debug_level is an integer which is formed by adding together the following values:

```
FUNCTION_TRACE      1
        ARGS_TRACE          2
        ASSIGNMENT_TRACE    4
        STATEMENT_TRACE     8
        MEMORY_TRACE        16
        TIMING_TRACE        32
        SUMMARY_TRACE       64
```

Ejemplo 1. apd_set_session() example

```
<?php
apd_set_session(9);
?>
```

apd_set_socket_session_trace

(no version information, might be only in CVS)

apd_set_socket_session_trace -- Starts the remote session debugging

Description

bool **apd_set_socket_session_trace** (string ip_address_or_unix_socket_file, int socket_type, int port, int debug_level)

Connects to the tcp server (eg. tcplisten) specified IP or Unix Domain socket (like a file), and sends debugging data to the socket. You can use any port, but higher numbers are better as most of the lower numbers may be used by other system services.

the socket_type can be APD_AF_UNIX (for file based sockets) or APD_AF_INET (for standard tcp/ip)

debug_level is an integer which is formed by adding together the following values:

```
FUNCTION_TRACE      1
        ARGS_TRACE          2
        ASSIGNMENT_TRACE    4
        STATEMENT_TRACE     8
        MEMORY_TRACE        16
        TIMING_TRACE        32
        SUMMARY_TRACE       64
```

I would seriously not recommend setting the value to 'zero' to start with, and use the breakpoint methods to start debugging at a specific place in the file.

Ejemplo 1. apd_set_socket_session_trace() example

```
<?php
  apd_set_socket_session_trace("127.0.0.1",APD_AF_INET,7112,0);
?>
```

override_function

(no version information, might be only in CVS)

override_function -- Overrides built-in functions

Description

bool **override_function** (string function_name, string function_args, string function_code)

Syntax similar to create_function(). Overrides built-in functions (replaces them in the symbol table).

Ejemplo 1. override_function() example

```
<?php
override_function('test', '$a,$b', 'echo "DOING TEST"; return $a * $b;');
?>
```

rename_function

(no version information, might be only in CVS)

rename_function -- Renames orig_name to new_name in the global function_table

Description

bool **rename_function** (string original_name, string new_name)

Renames orig_name to new_name in the global function_table. Useful for temporarily overriding builtin functions.

Ejemplo 1. rename_function() example

```
<?php
rename_function('mysql_connect', 'debug_mysql_connect' );
?>
```

III. Funciones de matrices

Introducción

Estas funciones permiten trabajar y manipular matrices (arrays) de diferentes maneras. Las matrices se utilizan para guardar, manejar y operar grupos de variables.

Matrices simples y multi-dimensionales están soportadas y pueden ser creadas por el usuario u otras funciones. Existen funciones específicas de manejo de bases de datos que actualizan matrices con el resultado devuelto por la base de datos, numerosas otras funciones devuelven matrices como resultado.

Consultar la sección del manual Matrices si quereis una explicación detallada de como las matrices están implementadas en PHP.

Requirimientos

No se necesitan bibliotecas externas para construir esta extensión

---

Instalación

No se necesita ninguna instalación para usar estas funciones, son parte del núcleo de PHP.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Las constantes listadas aqui están siempre disponibles a través del "núcleo PHP".

CASE_LOWER (integer)

> **CASE_LOWER** se utiliza con **array_change_key_case()** y se utiliza para convertir las claves de una matriz a minúsculas. Este es el valor por defecto de **array_change_key_case()**.

CASE_UPPER (integer)

> **CASE_UPPER** se utiliza con **array_change_key_case()** para convertir claves de matriz a mayúsculas.

Flags de ordenación:

SORT_ASC (integer)

> **SORT_ASC** se utiliza con **array_multisort()** para ordenar en sentido ascendente.

SORT_DESC (integer)

> **SORT_DESC** se utiliza con **array_multisort()** para ordenar en sentido descendente.

Flags de tipo ordenación: utilizadas por varias funciones ordenación

SORT_REGULAR (integer)

> **SORT_REGULAR** se utiliza para comparar elementos de forma normal.

SORT_NUMERIC (integer)

> **SORT_NUMERIC** se utiliza para comparar elementos de forma numérica.

SORT_STRING (integer)

> **SORT_STRING** se utiliza para comparar elementos como cadenas.

COUNT_NORMAL (**integer**)

COUNT_RECURSIVE (integer)

EXTR_OVERWRITE (integer)

EXTR_SKIP (integer)

EXTR_PREFIX_SAME (integer)

EXTR_PREFIX_ALL (integer)

EXTR_PREFIX_INVALID (integer)

EXTR_PREFIX_IF_EXISTS (integer)

EXTR_IF_EXISTS (integer)

EXTR_REFS (integer)

Ver también

Ver tambien **is_array()**, **explode()**, **implode()**, **split()**, **preg_split()** y **unset()**.

Tabla de contenidos

array_pop -- Extrae el último elemento de la matriz
array_push -- Inserta uno o más elementos al final de la matriz
array_rand -- Selecciona una o más entradas aleatorias de una matriz
array_reduce -- Reduce iterativamente una matriz a un solo valor usando una función llamada de retorno
array_reverse -- Devuelve una matriz con los elementos en orden inverso
array_search -- Busca un valor determinado en una matriz y devuelve la clave correspondiente en caso de éxito
array_shift -- Extrae un elemento del comienzo de la matriz
array_slice -- Extrae una porción de la matriz
array_splice -- Suprime una porción de la matriz y la sustituye por otra cosa
array_sum -- Calcula la suma de los valores en una matriz
array_udiff_assoc -- Computa la diferencia entre matrices con un chequeo de índices adicional, comparando los datos con una llamada de retorno
array_udiff_uassoc -- Computa la diferencia entre matrices con un chequeo de índices adicional, comparando los datos y los índices con una llamada de retorno
array_udiff -- Computa la diferencia entre matrices, usando una llamada de retorno para la comparación de datos
array_uintersect_assoc -- Calcula la intersección de matrices con chequeo adicional de ídices, comparando los datos por una función del usuario
array_uintersect_uassoc -- Calcula la intersección de matrices con chequeo adicional de índices, compara los datos y los índices por una función del usuario
array_uintersect -- Calcula la intersecćon de matrices, compara los datos con una función del usuario
array_unique -- Remueve valores duplicados de una matriz
array_unshift -- Introduce uno o más elementos al principio de la matriz
array_values -- Devuelve todos los valores de una matriz
array_walk_recursive -- Aplicar una función de usuario recursivamente a cada miembro de una matriz
array_walk -- Aplica una función del usuario a cada elemento de una matriz.
array -- Crear una matriz
arsort -- Ordena una matriz en orden inverso y mantiene la asociación de índices
asort -- Ordena una matriz y mantiene la asociación de índices
compact -- Crea una matriz que contiene variables y sus valores
count -- Cuenta los elementos de una matriz o propiedades de un objeto
current -- Devuelve el elemento actual de una matriz
each -- Devuelve el siguiente par clave/valor de una matriz y avanza el apuntador
end -- Mueve el puntero interno de una tabla al último elemento
extract -- Importa variables a la tabla de símbolos desde una matriz
in_array -- Revisa si un valor existe en una matriz
key -- Obtiene una clave de una matriz asociativa
krsort -- Ordena una matriz por clave en orden inverso
ksort -- Ordena una matriz por clave
list -- Asigna variables como si fueran una matriz
natcasesort -- Ordena una matriz usando un algoritmo de "orden natural" sin distinguir mayúsculas de minúsculas
natsort -- Ordena una matriz usando un algoritmo de "orden natural"
next -- Avanza el puntero interno de una matriz
pos -- Alias de **current()**
prev -- Rebobina el puntero interno de una matriz
range -- Crea una matriz que contiene un rango de elementos
reset -- Fija el puntero interno de una matriz a su primer elemento
rsort -- Ordena una matriz en orden inverso
shuffle -- Mezcla una matriz
sizeof -- Alias de **count()**

sort -- Ordena una matriz
uasort -- Ordena una matriz mediante una función de comparación definida por el usuario y mantiene la asociación de índices
uksort -- Ordena una matriz por claves mediante una función definida por el usuario
usort -- Ordena una matriz por sus valores usando una función de comparación definida por el usuario

array_change_key_case

(PHP 4 >= 4.2.0, PHP 5)

array_change_key_case -- Devuelve una matriz con todas las claves de las cadenas convertidas a mayúsculas o minúsculas

Descripción

array **array_change_key_case** (array input [, int case])

array_change_key_case() cambia las claves de la matriz *input* de forma que éstas sean todas mayúsculas o minúsculas. El cambio depende del parámetro opcional *case*, mediante el cual se pueden pasar dos constantes a la función: **CASE_UPPER** (mayúsculas) y **CASE_LOWER** (minúsculas). El valor por defecto es **CASE_LOWER**. La función no afectará a los índices numéricos.

Ejemplo 1. Ejemplo de array_change_key_case()

```
$input_array = array("PriMero" => 1, "SeGunDo" => 4);
print_r(array_change_key_case($input_array, CASE_UPPER));
```

La salida del programa anterior será:

```
Array
(
    [PRIMERO] => 1
    [SEGUNDO] => 4
)
```

array_chunk

(PHP 4 >= 4.2.0, PHP 5)

array_chunk -- Divide una matriz en segmentos

Descripción

array **array_chunk** (array input, int size [, bool preserve_keys])

array_chunk() divide una matriz en varias matrices, cada una con un número de valores ñalente a *size*. La última matriz generada puede tener un número menor de valores. Las matrices se obtienen como miembros de una matriz multidimensional, que será indexada con números empezando por el cero.

Si al parámetro opcional *preserve_keys* se le da el valor de **TRUE**, se fuerza a PHP a preservar las claves originales de la matriz de entrada. Si se especifica **FALSE**, se utilizarán nuevos índices numéricos en cada matriz resultante. El valor por defecto es **FALSE**.

Ejemplo 1. Ejemplo de array_chunk()

```
$input_array = array('a', 'b', 'c', 'd', 'e');
print_r(array_chunk($input_array, 2));
print_r(array_chunk($input_array, 2, TRUE));
```

La salida del anterior programa será:

```
Array
(
    [0] => Array
        (
            [0] => a
            [1] => b
        )

    [1] => Array
        (
            [0] => c
            [1] => d
        )

    [2] => Array
        (
            [0] => e
        )

)
Array
(
    [0] => Array
        (
            [0] => a
            [1] => b
        )

    [1] => Array
        (
            [2] => c
            [3] => d
        )

    [2] => Array
        (
            [4] => e
        )

)
```

array_combine

(PHP 5)

array_combine -- Crea una nueva matriz, usando una matriz para las claves y otra para sus valores

Descripción

array **array_combine** (array claves, array valores)

Devuelve un **array** usando los valores de la matriz *claves* como claves, y los valores de la matriz *valores* como los valores correspondientes.

Devuelve **FALSE** si el número de elementos de cada matriz no es ñalente o si las matrices están vacías.

Ejemplo 1. Un ejemplo simple de array_combine()

```
<?php
$a = array('verde', 'rojo', 'amarillo');
$b = array('aguacate', 'manzana', 'banano');
$c = array_combine($a, $b);

print_r($c);
?>
```

Genera la salida:

```
Array
(
    [verde]    => aguacate
    [rojo]     => manzana
    [amarillo] => banano
)
```

Vea también **array_merge()**, **array_walk()**, y **array_values()**.

array_count_values

(PHP 4 , PHP 5)

array_count_values -- Cuenta todos los valores de una matriz

Descripción

array **array_count_values** (array entrada)

array_count_values() devuelve una matriz usando los valores de la matriz *entrada* como índices y su frecuencia de aparición en la *entrada* como valores.

Ejemplo 1. Ejemplo de array_count_values()

```
<?php
$matriz = array(1, "hola", 1, "mundo", "hola");
array_count_values($matriz); // devuelve array(1=>2, "hola"=>2, "mundo"=>1)
?>
```

El resultado del ejemplo seria:

```
Array
(
    [1] => 2
    [hola] => 2
    [mundo] => 1
)
```

Vea también **count()**, **array_unique()**, **array_values()**, **count_chars()**.

array_diff_assoc

(PHP 4 >= 4.3.0, PHP 5)

array_diff_assoc -- Comprueba las diferencias entre matrices teniendo en cuenta los índices

Descripción

array **array_diff_assoc** (array array1, array array2 [, array ...])

array_diff_assoc() devuelve una matriz que contiene todos los valores de *array1* que no estén presentes en ninguna de las otras matrices que se pasan como argumento. Hay que tener en cuenta que las claves de los valores se utilizan en la comparación, a diferencia de **array_diff()**.

Ejemplo 1. Ejemplo de array_diff_assoc()

```
<?php
$array1 = array ("a" => "verde", "b" => "negro", "c" => "azul", "rojo");
$array2 = array ("a" => "verde", "amarillo", "rojo");
$result = array_diff_assoc ($array1, $array2);

/* El resultado es:
Array
(
    [b] => negro
    [c] => azul
    [0] => rojo
)
*/
?>
```

En el ejemplo anterior se ve que el par *"a" => "verde"* está presente en ambas matrices y por ello no aparece en la matriz resultado. Por el contrario, el par *0 => "rojo"* sí aparece en el resultado puesto que en el segundo argumento el valor *"red"* tiene la clave *1*.

Dos valores de pares *clave => valor* se consideran iguales si y sólo si *(string) $elem1 === (string) $elem2*, es decir, cuando la representación de cadena es la misma.

> **Nota:** Hay que tener en cuenta que esta función sólo comprueba una dimensión de una matriz n-dimensional. Por supuesto, se pueden comprobar dimensiones sucesivas mediante: *array_diff_assoc($array1[0], $array2[0]);*.

Véase también **array_diff()**, **array_intersect()**, y **array_intersect_assoc()**.

array_diff_key

(no version information, might be only in CVS)

array_diff_key -- Calcula la diferencia de matrices usando las llaves para la comparación

Descripción

array **array_diff_key** (array array1, array array2 [, array ...])

array_diff_key() regresa una matriz conteniendo todos los valores de *array1* que tienen llaves que no están presentes en cualquier otra matriz dada como parámetro. Note que la asociatividad es preservada. Esta función es como **array_diff()** excepto en que la comparación es hecha en las llaves en lugar de en los valores.

Ejemplo 1. Ejemplo array_diff_key()

```
<?php
$array1 = array('blue'  => 1, 'red'  => 2, 'green'  => 3, 'purple' => 4);
$array2 = array('green' => 5, 'blue' => 6, 'yellow' => 7, 'cyan'   => 8);

var_dump(array_diff_key($array1, $array2));
?>
```

El resultado del ejemplo seria:

```
array(2) {
  ["red"]=>
  int(2)
  ["purple"]=>
  int(4)
}
```

Las dos llaves del par *llave => valor*, son consideradas iguales sólo si *(string) $key1 === (string) $key2* . En otras palabras se revisa estrictamente el tipo de dato de tal manera que la representación del string debe ser el mismo.

> **Nota:** Note por favor que esta función solo revisa una dimensión de una matriz con n dimensiones. Por supuesto puede checar en forma más profunda usando la forma *array_diff_key($array1[0], $array2[0]);*.

Vea también **array_diff()**, **array_udiff()** **array_diff_assoc()**, **array_diff_uassoc()**, **array_udiff_assoc()**, **array_udiff_uassoc()**, **array_diff_ukey()**, **array_intersect()**, **array_intersect_assoc()**, **array_intersect_uassoc()**, **array_intersect_key()** y **array_intersect_ukey()**.

array_diff_uassoc

(PHP 5)

array_diff_uassoc -- Computa la diferencia entre matrices con un chequeo adicional de índices, el cual es realizado por una llamada de retorno entregada por el usuario

Descripción

array **array_diff_uassoc** (array matriz1, array matriz2 [, array ..., callback func_comparacion_claves])

array_diff_uassoc() devuelve un **array** que contiene todos los valores de *matriz1* que no están presentes en ninguno de los otros argumentos. Note que las claves son usadas en la comparación, a diferencia de **array_diff()**. Esta comparación es realizada por una llamada de retorno entregada por el usuario. La función debe devolver un entero menor que, igual, o mayor que cero si el primer argumento es considerado como menor, igual, o mayor que el segundo, respectivamente. Esto en contraste a **array_diff_assoc()** en donde es usada una función interna para la comparación de índices.

Ejemplo 1. Ejemplo de array_diff_uassoc()

```
<?php
function func_comparacion_claves($a, $b)
{
    if ($a === $b) {
        return 0;
    }
    return ($a > $b)? 1:-1;
}

$matriz1 = array("a" => "green", "b" => "brown", "c" => "blue", "red");
$matriz2 = array("a" => "green", "yellow", "red");
$resultado = array_diff_uassoc($matriz1, $matriz2, "func_comparacion_claves");
?>
```

El resultado es:

```
Array
(
    [b] => brown
    [c] => blue
    [0] => red
)
```

En nuestro ejemplo anterior, puede ver que la pareja *"a" => "green"* está presente en ambas matrices y por lo tanto no hace parte de la salida de la función. En contraste, la pareja *0 => "red"* está en la salida ya que en el segundo argumento, *"red"* tiene una clave que es *1*.

La igualdad de 2 índices es revisada por la llamada de retorno indicada por el usuario.

> **Nota:** Por favor note que esta función sólo analiza una dimensión de una matriz n-dimensional. Por supuesto que puede analizar dimensiones más profundas usando, por ejemplo, *array_diff_uassoc($matriz1[0], $matriz2[0], "func_comparacion_claves");*.

Vea también **array_diff()**, **array_diff_assoc()**, **array_udiff()**, **array_udiff_assoc()**, **array_udiff_uassoc()**, **array_intersect()**, **array_intersect_assoc()**, **array_uintersect()**, **array_uintersect_assoc()** y **array_uintersect_uassoc()**.

array_diff_ukey

(no version information, might be only in CVS)

array_diff_ukey -- Calcula la diferencia de matrices usando callback function on the keys for comparison

Descripción

array **array_diff_ukey** (array array1, array array2 [, array ..., callback key_compare_func])

array_diff_ukey() regresa una matriz conteniendo todos los valores de *array1* que tienen llaves que no están presentes en cualquiera de los otros argumentos. Note que la asociatividad es preservada. Esta función es como **array_diff()** excepto en que la comparación es hecha en las llaves en ligar de en los valores.

Esta comparación es hecha por una función proveída por el usuario. Debe regresar un entero menor que, igual a, o mayor que cero si la primera llave es considerada respectivamente menor que, igual a, o mayor que la segunda llave.

Ejemplo 1. Ejemplo array_diff_ukey()

```
<?php
function key_compare_func($key1, $key2)
{
    if ($key1 == $key2)
        return 0;
    else if ($key1 > $key2)
        return 1;
    else
        return -1;
}

$array1 = array('blue'  => 1, 'red'  => 2, 'green'  => 3, 'purple' => 4);
$array2 = array('green' => 5, 'blue' => 6, 'yellow' => 7, 'cyan'   => 8);

var_dump(array_diff_ukey($array1, $array2, 'key_compare_func'));
?>
```

El resultado del ejemplo seria:

```
array(2) {
  ["red"]=>
  int(2)
  ["purple"]=>
  int(4)
}
```

Las dos llaves de los pares *llave => valor*, son consideradas iguales solo si *(string) $key1 === (string) $key2* . En otras palabras se hace un estricto chequeo de tipo de tal manera que la representación de la cadena debe ser la misma.

> **Nota:** Note por favor que esta función solo revisa una dimensión en una matriz con n dimensiones. Por supuesto puede revisar más profundamente usando la forma *array_diff_ukey($array1[0], $array2[0], 'callback_func');*.

Vea tambié **array_diff()**, **array_udiff()** **array_diff_assoc()**, **array_diff_uassoc()**, **array_udiff_assoc()**, **array_udiff_uassoc()**, **array_diff_key()**, **array_intersect()**, **array_intersect_assoc()**, **array_intersect_uassoc()**, **array_intersect_key()** y **array_intersect_ukey()**.

array_diff

(PHP 4 >= 4.0.1, PHP 5)

array_diff -- Comprueba las diferencias entre matrices

Descripción

array **array_diff** (array array1, array array2 [, array ...])

array_diff() devuelve una matriz que contiene todos los valores de *array1* que no aparezcan en ninguna de las otras matrices que se pasan a la función como argumento. Hay que tener en cuenta que las claves se mantienen.

Ejemplo 1. Ejemplo de array_diff()

```
$array1 = array ("a" => "verde", "rojo", "azul", "rojo");
$array2 = array ("b" => "verde", "amarillo", "rojo");
$resultado = array_diff ($array1, $array2);
```

Lo cual hace que *$resultado* contenga la matriz *array ("azul");*. Multiples ocurrencias en $array1 son tratadas todas de la misma forma.

> **Nota:** Dos elementos se consideran ñalentes si y sólo si *(string) $elem1 === (string) $elem2*, es decir, cuando la representación de cadena es la misma.

> **Nota:** Hay que tener en cuenta que esta función sólo comprueba una dimensión de una matriz n-dimensional. Por supuesto, se pueden comprobar dimensiones sucesivas mediante: *array_diff($array1[0], $array2[0]);*.

Aviso
Esta función se rompió en PHP 4.0.4!

Véase también **array_diff_assoc()**, **array_intersect()** y **array_intersect_assoc()**.

array_fill

(PHP 4 >= 4.2.0, PHP 5)

array_fill -- Llena una matriz con valores

Descripción

array **array_fill** (int start_index, int num, mixed value)

array_fill() introduce en una matriz tantas entradas como especifique el parámetro *num*, numerandolas a partir de *start_index*, y les asigna el valor del parámetro *value*.

Ejemplo 1. Ejemplo de array_fill()

```
$a = array_fill(5, 6, 'banana');
print_r($a);
```

$a contendrá ahora:

```
Array
(
    [5]  => banana
    [6]  => banana
    [7]  => banana
    [8]  => banana
    [9]  => banana
    [10] => banana
)
```

array_filter

(PHP 4 >= 4.0.6, PHP 5)

array_filter -- Filtra elementos de una matriz mediante una función "callback"

Descripción

array **array_filter** (array input [, callback function])

array_filter() repasa cada valor en la matriz *input* y lo pasa a la función *callback*. Si la función *callback* devuelve verdadero, el valor es devuelto a la matriz resultado. Los índices de la matriz se mantienen.

Ejemplo 1. Ejemplo de array_filter()

```
<?php
function impar($var) {
    return ($var % 2 == 1);
}

function par($var) {
    return ($var % 2 == 0);
}

$array1 = array ("a"=>1, "b"=>2, "c"=>3, "d"=>4, "e"=>5);
$array2 = array (6, 7, 8, 9, 10, 11, 12);

echo "Impares :\n";
print_r(array_filter($array1, "impar"));
echo "Pares:\n";
print_r(array_filter($array2, "par"));
?>
```

La salida en pantalla del programa anterior será:

```
Impares :
Array
(
    [a] => 1
    [c] => 3
    [e] => 5
)
Pares:
Array
(
    [0] => 6
    [2] => 8
    [4] => 10
    [6] => 12
)
```

No se debe modificar la matriz a la que **array_filter()** se aplica desde la función "callback", como p. ej. añadir o eliminar un elemento, o realizar un "unset()". Si la matriz es modificada el comportamiento de esta función no está definido.

Ver también **array_map()**, **array_reduce()**, y **array_walk()**.

array_flip

(PHP 4 , PHP 5)

array_flip -- Intercambia los valores de una matriz con sus índices

Descripción

array **array_flip** (array trans)

array_flip() devuelve una matriz con los valores intercambiados, por ejemplo los índices de *trans* se convierten en los valores y los valores de *trans* se convierten en los índices.

note que los valores de *trans* necesitan ser índices validos, eg. necesitan ser del tipo **integer** o **string**. Se generará una alerta si un valor tiene un tipo diferente, y el par índice/valor en cuestión *no será modificado*.

Si un valor se encuentra varias veces, el último índice será usado con su valor, y todos los demás se perderán

array_flip() regresa **FALSE** su falla.

Ejemplo 1. Ejemplo de array_flip()

```
<?php
$trans = array_flip($trans);
$original = strtr($str, $trans);
?>
```

Ejemplo 2. Ejemplo de colisión con array_flip()

```
<?php
$trans = array("a" => 1, "b" => 1, "c" => 2);
$trans = array_flip($trans);
print_r($trans);
?>
```

ahora *$trans* es:

```
Array
(
    [1] => b
    [2] => c
)
```

Vea también **array_values()**, **array_keys()**, y **array_reverse()**.

array_intersect_assoc

(PHP 4 >= 4.3.0, PHP 5)

array_intersect_assoc -- Computes the intersection of arrays with additional index check

Description

array **array_intersect_assoc** (array array1, array array2 [, array ...])

array_intersect_assoc() returns an array containing all the values of *array1* that are present in all the arguments. Note that the keys are used in the comparison unlike in **array_intersect()**.

Ejemplo 1. array_intersect_assoc() example

```
<?php
$array1 = array("a" => "green", "b" => "brown", "c" => "blue", "red");
$array2 = array("a" => "green", "yellow", "red");
$result_array = array_intersect_assoc($array1, $array2);
?>
```

$result_array will look like:

```
Array
(
    [a] => green
)
```

In our example you see that only the pair *"a" => "green"* is present in both arrays and thus is returned. The value *"red"* is not returned because in *$array1* its key is *0* while the key of "red" in *$array2* is *1*.

The two values from the *key => value* pairs are considered equal only if *(string) $elem1 === (string) $elem2* . In other words a strict type check is executed so the string representation must be the same.

See also **array_intersect()**, **array_uintersect_assoc()**, **array_intersect_uassoc()**, **array_uintersect_uassoc()**, **array_diff()** and **array_diff_assoc()**.

array_intersect_key

(no version information, might be only in CVS)

array_intersect_key -- Calcula la intersección de matrices usando las llaves para la comparación

Descripción

array **array_intersect_key** (array array1, array array2 [, array ...])

array_intersect_key() regresa una matriz conteniendo todos los valores de *array1* los cuales tienen llaves que están presentes en todos los argumentos.

Ejemplo 1. Ejemplo array_intersect_key()

```
<?php
$array1 = array('azul'  => 1, 'rojo'  => 2, 'verde'  => 3, 'morado' => 4);
$array2 = array('verde' => 5, 'azul' => 6, 'amarillo' => 7, 'cyan'   => 8);

var_dump(array_intersect_key($array1, $array2));
?>
```

El resultado del ejemplo seria:

```
array(2) {
  ["azul"]=>
  int(1)
  ["verde"]=>
  int(3)
})
```

En nuestro ejemplo puede ver que sólo las llaves *'azul'* y *'verde'* están presentes en ambas matrices y por lo tanto son regresadas. TambiÃ‚Â´n note que los valores para las llaves *'azul'* y *'verde'* son diferentes en las dos matrices. Aún así siguen coincidiendo porque se hace la comparación sólo en las llaves. Los valores regresados son aquellos de *array1*.

Las dos llaves del par *key => value*, son considerados iguales solo si *(string) $key1 === (string) $key2* . En otras palabras se hace un chequeo estricto del tipo de dato de tal manera que la representación de la cadena debe ser la misma.

Vea también **array_diff()**, **array_udiff()** **array_diff_assoc()**, **array_diff_uassoc()**, **array_udiff_assoc()**, **array_udiff_uassoc()**, **array_diff_key()**, **array_diff_ukey()**, **array_intersect()**, **array_intersect_assoc()**, **array_intersect_uassoc()** y **array_intersect_ukey()**.

array_intersect_uassoc

(PHP 5)

array_intersect_uassoc -- Calcula la intersección de matrices con chequeo de índices adicional por una función de usuario

Descripción

array **array_intersect_uassoc** (array array1, array array2 [, array ..., callback key_compare_func])

array_intersect_uassoc() Regresa una matriz conteniendo todos los valores de *array1* que están presentes en todos los argumentos. Note que los índices son usados en la comparación a diferencia

de **array_intersect()**.

La comparación de índices es hecha por una función de usuario. Esta debe regresar un entero menor que, igual a, o mayor que cero si el primer argumento es considerado ser menor que, igual a o mayor que el segundo, respectivamente.

Ejemplo 1. Ejemplo de array_intersect_uassoc()

```
<?php
$array1 = array("a" => "green", "b" => "brown", "c" => "blue", "red");
$array2 = array("a" => "GREEN", "B" => "brown", "yellow", "red");

print_r(array_intersect_uassoc($array1, $array2, "strcasecmp"));
?>
```

El resultado del ejemplo seria:

```
Array
(
    [b] => brown
)
```

Vea también: **array_intersect()**, **array_intersect_assoc()**, **array_uintersect_assoc()**, **array_uintersect_uassoc()**, **array_intersect_key()** y **array_intersect_ukey()**.

array_intersect_ukey

(no version information, might be only in CVS)

array_intersect_ukey -- Calcula la intersecció de matrices usando una función de usuario para la comparación de los índices

Descripción

array **array_intersect_ukey** (array array1, array array2 [, array ..., callback key_compare_func])

array_intersect_ukey() regresa una matriz que contiene todos los valores de *array1* los cuales tienen índices que están presentes en todos los argumentos.

Esta comparación es hecha por una función del usuario. La cual debe regresar un entero menor que, igual a, o mayor que cero si el primer índice es considerado ser menor que, igual a o mayor que el segundo, respectivamente.

Ejemplo 1. Ejemplo array_intersect_ukey()

```
<?php
function key_compare_func($key1, $key2)
{
    if ($key1 == $key2)
        return 0;
    else if ($key1 > $key2)
        return 1;
    else
        return -1;
}

$array1 = array('blue'  => 1, 'red'  => 2, 'green'  => 3, 'purple' => 4);
$array2 = array('green' => 5, 'blue' => 6, 'yellow' => 7, 'cyan'   => 8);

var_dump(array_intersect_ukey($array1, $array2, 'key_compare_func'));
?>
```

El resultado del ejemplo seria:

```
array(2) {
  ["blue"]=>
  int(1)
  ["green"]=>
  int(3)
})
```

En nuestro ejemplo ve que solo los índices *'blue'* y *'green'* están presentes en ambas matrices y por lo tanto son regresados. También note los valores de los índices *'blue'* y *'green'* son diferentes entre las dos matrices. Aún así coinciden porque solo los índices son checados. Los valores regresados son los de *array1*.

Los dos índices del par *key => value* son considerados igual solo si *(string) $key1 === (string) $key2* . En otras palabras se hace un chequeo estricto de tipo, de tal manera que la representación de la cadena sea la misma.

Vea también: **array_diff()**, **array_udiff()** **array_diff_assoc()**, **array_diff_uassoc()**, **array_udiff_assoc()**, **array_udiff_uassoc()**, **array_diff_key()**, **array_diff_ukey()**, **array_intersect()**, **array_intersect_assoc()**, **array_intersect_uassoc()** y **array_intersect_key()**.

array_intersect

(PHP 4 >= 4.0.1, PHP 5)

array_intersect -- Computes the intersection of arrays

Description

array **array_intersect** (array array1, array array2 [, array ...])

array_intersect() returns an array containing all the values of *array1* that are present in all the arguments. Note that keys are preserved.

Ejemplo 1. array_intersect() example

```
<?php
$array1 = array("a" => "green", "red", "blue");
$array2 = array("b" => "green", "yellow", "red");
$result = array_intersect($array1, $array2);
?>
```

This makes *$result* have

```
Array
(
    [a] => green
    [0] => red
)
```

> **Nota:** Two elements are considered equal if and only if *(string) $elem1 === (string) $elem2*. In words: when the string representation is the same.

See also **array_intersect_assoc()**, **array_diff()**, and **array_diff_assoc()**.

array_key_exists

(PHP 4 >= 4.1.0, PHP 5)

array_key_exists -- Comprueba si el índice o clave dada existe en la matriz

Descripción

bool **array_key_exists** (mixed clave, array fuente)

array_key_exists() devuelve **TRUE** si la *clave* dada existe en la matriz. La *clave* puede ser cualquier valor válido como índice de una matriz. **array_key_exists()** funciona también con objetos.

Ejemplo 1. Ejemplo de array_key_exists()

```
<?php
$matriz_a_buscar = array('primero' => 1, 'segundo' => 4);
if (array_key_exists('primero', $matriz_a_buscar)) {
    echo "El elemento 'primero' se encuentra en la matriz";
}
?>
```

Nota: El nombre de esta función es **key_exists()** en PHP 4.0.6.

Ejemplo 2. array_key_exists() vs isset()

isset() no devuelve **TRUE** para claves de matriz que corresponden a un valor **NULL**, mientras que **array_key_exists()** lo hace.

```
<?php
$matriz_busqueda = array('primero' => null, 'segundo' => 4);

// devuelve false
isset($matriz_busqueda['primero'])

// devuelve true
array_key_exists('primero', $matriz_busqueda);
?>
```

Vea también **isset()**, **array_keys()**, y **in_array()**.

array_keys

(PHP 4 , PHP 5)

array_keys -- Devuelve todas las claves de una matriz

Descripción

array **array_keys** (array entrada [, mixed val_a_buscar [, bool strict]])

array_keys() devuelve las claves, numéricas y de cadena, de la matriz *entrada*.

Si se especifica el parámetro opcional *val_a_buscar*, sólo se devuelven las claves para dicho valor. De otro modo, se devuelven todas las claves de la *entrada*. En PHP 5, puede usar el parámetro *strict* para comparaciones incluyendo el tipo (===).

Ejemplo 1. Ejemplo de array_keys()

```
<?php
$array = array(0 => 100, "color" => "red");
print_r(array_keys($array));

$array = array("blue", "red", "green", "blue", "blue");
print_r(array_keys($array, "blue"));

$array = array("color" => array("blue", "red", "green"),
               "size"  => array("small", "medium", "large"));
print_r(array_keys($array));
?>
```

El resultado del ejemplo seria:

```
Array
(
    [0] => 0
    [1] => color
)
Array
(
    [0] => 0
    [1] => 3
    [2] => 4
)
Array
(
    [0] => color
    [1] => size
)
```

Vea también: **array_values()**, **array_key_exists()**.

array_map

(PHP 4 >= 4.0.6, PHP 5)

array_map -- Aplica la llamada de retorno especificada a los elementos de las matrices dadas

Descripción

array **array_map** (callback llamada_de_retorno, array matriz1 [, array ...])

array_map() devuelve una matriz que contiene todos los elementos de *matriz1* después de haber aplicado la función *llamada_de_retorno* a cada uno de ellos. El número de parámetros que la función *llamada_de_retorno* acepte debería coincidir con el número de matrices que son pasadas como argumentos a **array_map()**.

Ejemplo 1. Ejemplo de array_map()

```
<?php
function cubo($n)
{
    return($n * $n * $n);
}

$a = array(1, 2, 3, 4, 5);
$b = array_map("cubo", $a);
print_r($b);
?>
```

Esto hace que *$b* contenga:

```
Array
(
    [0] => 1
    [1] => 8
    [2] => 27
    [3] => 64
    [4] => 125
)
```

Ejemplo 2. array_map() - usando más matrices

```
<?php
function mostrar_Castellano($n, $m)
{
    return("El n&uacute;mero $n es llamado $m en Castellano");
}

function map_Castellano($n, $m)
{
    return(array($n => $m));
}

$a = array(1, 2, 3, 4, 5);
$b = array("uno", "dos", "tres", "cuatro", "cinco");

$c = array_map("mostrar_Castellano", $a, $b);
print_r($c);

$d = array_map("map_Castellano", $a , $b);
print_r($d);
?>
```

Esto produce:

```
// salida correspondiente a $c
Array
(
    [0] => El n&uacute;mero 1 es llamado uno en Castellano
    [1] => El n&uacute;mero 2 es llamado dos en Castellano
    [2] => El n&uacute;mero 3 es llamado tres en Castellano
    [3] => El n&uacute;mero 4 es llamado cuatro en Castellano
    [4] => El n&uacute;mero 5 es llamado cinco en Castellano
)

// salida correspondiente a $d
Array
(
    [0] => Array
        (
            [1] => uno
        )

    [1] => Array
        (
            [2] => dos
        )

    [2] => Array
        (
            [3] => tres
        )

    [3] => Array
        (
            [4] => cuatro
        )

    [4] => Array
        (
            [5] => cinco
        )

)
```

Usualmente, cuando se usan dos o más matrices, éstas deberían ser de longitudes iguales ya que la llamada de retorno es aplicada en paralelo a los elementos correspondientes. Si las matrices son de longitudes diferentes, la más corta de ellas será extendida con elementos vacíos.

Un uso interesante de esta función es la construcción de una matriz de matrices, que puede ser llevada a cabo usando **`NULL`** como el nombre de la llamada de retorno.

Ejemplo 3. Creación de una matriz de matrices

```
<?php
$a = array(1, 2, 3, 4, 5);
$b = array("one", "two", "three", "four", "five");
$c = array("uno", "dos", "tres", "cuatro", "cinco");

$d = array_map(null, $a, $b, $c);
print_r($d);
?>
```

La salida del anterior programa será:

```
Array
(
    [0] => Array
        (
            [0] => 1
            [1] => one
            [2] => uno
        )

    [1] => Array
        (
            [0] => 2
            [1] => two
            [2] => dos
        )

    [2] => Array
        (
            [0] => 3
            [1] => three
            [2] => tres
        )

    [3] => Array
        (
            [0] => 4
            [1] => four
            [2] => cuatro
        )

    [4] => Array
        (
            [0] => 5
            [1] => five
            [2] => cinco
        )

)
```

Vea también **array_filter()**, **array_reduce()**, y **array_walk()**.

array_merge_recursive

(PHP 4 >= 4.0.1, PHP 5)

array_merge_recursive -- Une dos o más matrices recursivamente

Descripción

array **array_merge_recursive** (array matriz1, array matriz2 [, array ...])

array_merge_recursive() une los elementos de dos o más matrices de modo tal que los valores de cada una sean añadidos al final de la matriz previa. Devuelve la matriz resultante.

Si las matrices de entrada tienen las mismas claves tipo cadena, entonces los valores de estas claves son unidas en una matriz, y esto es hecho recursivamente, de modo que si uno de los valores es una matriz misma, la función unirá también ésta con la correspondiente entrada de otra matriz. Sin embargo, si las matrices tienen la misma clave numérica, el valor más cercano al final no sobrescribirá el valor original, sino que será añadido al final.

Ejemplo 1. Ejemplo de array_merge_recursive()

```
<?php
$m1 = array("color" => array("favorito" => "rojo"), 5);
$m2 = array(10, "color" => array("favorito" => "verde", "azul"));
$resultado = array_merge_recursive($m1, $m2);
?>
```

El *$resultado* será:

```
Array
(
    [color] => Array
        (
            [favorito] => Array
                (
                    [0] => rojo
                    [1] => verde
                )

            [0] => azul
        )

    [0] => 5
    [1] => 10
)
```

Vea también **array_merge()**.

array_merge

(PHP 4 , PHP 5)

array_merge -- Combina dos o más matrices

Descripción

array **array_merge** (array matriz1 [, array matriz2 [, array ...]])

array_merge() combina los elementos de dos o más matrices conjuntamente de modo que los valores de una son agregados al final de los valores de la anterior. Devuelve la matriz resultante.

Si las matrices de entrada tienen las mismas claves de cadena, el último valor para cada clave reemplazará el valor previo de la misma. Si, por el contrario, las matrices tienen la misma clave numérica, esto no pasa y los valores son simplemente agregados.

Si sólo se da una matriz, y si tiene índices numéricos, los índices son reacomodados. Para matrices asociativas, los elementos duplicados se combinarán en el último. Vea el ejemplo tres para más detalles.

Ejemplo 1. Ejemplo de array_merge()

```
<?php
$array1 = array("color" => "red", 2, 4);
$array2 = array("a", "b", "color" => "green", "shape" => "trapezoid", 4);
$result = array_merge($array1, $array2);
print_r($result);
?>
```

El *$result* es:

```
Array
(
    [color] => green
    [0] => 2
    [1] => 4
    [2] => a
    [3] => b
    [shape] => trapezoid
    [4] => 4
)
```

Ejemplo 2. Ejemplo simple de array_merge()

```
<?php
$array1 = array();
$array2 = array(1 => "data");
$result = array_merge($array1, $array2);
?>
```

No olvide que los índices numéricos serán reenumerados

```
Array
(
    [0] => data
)
```

Si quiere preservar completamente las matrices y solo quiere agregarlos unos a otros, use el operador + :

```
<?php
$array1 = array();
$array2 = array(1 => "data");
$result = $array1 + $array2;
?>
```

Los índices numéricos son preservados y por lo tanto la asocuación permanece.

```
Array
(
    [1] => data
)
```

Ejemplo 3. Ejemplo array_merge()

```
<?php
$array_one = array(0 => "jay", 1 => "bob", 2 => "randal", 3 => "dante");
$array_two = array("jay" => "bob", "randal" => "dante", "jay" => "jason");

unset($array_one[2]);

$result_one = array_merge($array_one);
$result_two = array_merge($array_two);

print_r($result_one);
print_r($result_two);
?>
```

El resultado del ejemplo seria:

```
Array
(
    [0] => jay
    [1] => bob
    [2] => dante
)
Array
(
    [jay] => jason
    [randal] => dante
)
```

> **Nota:** Los índices compartidos serán sobreescritos en base a la forma "primero en llegar, primero en salir".

Aviso

El comportamient de **array_merge()** fue modificado en PHP 5. A diferencia de PHP 4, la función **array_merge()** ahora acepta solo parámetros de tipo **array**. Sin embargo, puede usar forzado de tipos para convertir otros tipos. Vea el siguiente ejemplo para más detalles.

Ejemplo 4. Ejemplo PHP 5 array_merge()

```
<?php
$beginning = 'foo';
$end = array(1 => 'bar');
$result = array_merge((array)$beginning, (array)$end);
?>
```

El resultado del ejemplo seria:

```
Array
(
    [0] => foo
    [1] => bar
)
```

Vea también **array_merge_recursive()**, **array_combine()**, array operators.

array_multisort

(PHP 4 , PHP 5)

array_multisort -- Ordena múltiples matrices, o matrices multi-dimensionales

Descripción

bool **array_multisort** (array matriz1 [, mixed arg [, mixed ... [, array ...]]])

array_multisort() puede usarse para ordenar varias matrices al tiempo o una matriz multi-dimensional de acuerdo a una de sus varias dimensiones. Las llaves asociativas (tipo cadena) son conservadas, mientras que las llaves numéricas son re-indexadas.

Las matrices de entrada son tratadas como columnas de una tabla que deberá ser ordenada por filas - de forma similar a la funcionalidad de una sentencia SQL ORDER BY. La primera matriz es considerada la primaria para el ordenamiento. Las filas (valores) en esa matriz que sean comparadas como iguales son ordenadas por la siguiente matriz de entrada, y así sucesivamente.

La estructura de argumentos de esta función es un poco inusual, pero flexible. El primer argumento de todos debe ser una matriz. Subsecuentemente, cada argumento puede ser o una matriz o una bandera de ordenamiento de las siguientes.

Banderas de orientación del ordenamiento:

- **SORT_ASC** - ordenar ascendentemente
- **SORT_DESC** - ordenar descendentemente

Banderas de tipo de ordenamiento

- **SORT_REGULAR** - comparar elementos normalmente
- **SORT_NUMERIC** - comparar elementos numéricamente

- **SORT_STRING** - comparar elementos como cadenas

No pueden especificarse dos banderas de ordenamiento del mismo tipo luego de cada matriz. Las banderas de ordenamiento especificadas a continuación de un argumento matriz se aplican sólo a esa matriz - estos valores son restablecidos de vuelta a **SORT_ASC** y **SORT_REGULAR** antes de cada nuevo argumento matriz.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplo 1. Ordenamiento de varias matrices

```
<?php
$matriz1 = array("10", 100, 100, "a");
$matriz2 = array(1, 3, "2", 1);
array_multisort($matriz1, $matriz2);
?>
```

En este ejemplo, después del ordenamiento, la primera matriz contendrá los valores 10, "a", 100, 100. La segunda matriz contendrá 1, 1, "2", 3. Las entradas en la segunda matriz que correspondían a las entradas idénticas de la primera matriz (100 y 100) fueron ordenadas también.

Ejemplo 2. Ordenamiento de una matriz multi-dimensional

```
<?php
$matriz = array(array("10", 100, 100, "a"), array(1, 3, "2", 1));
array_multisort($matriz[0], SORT_ASC, SORT_STRING,
                $matriz[1], SORT_NUMERIC, SORT_DESC);
?>
```

En este ejemplo, después del ordenamiento, la primera matriz contendrá 10, 100, 100, "a" (fue ordenada como cadenas en orden ascendente), y la segunda tendrá 1, 3, "2", 1 (ordenada como números, en orden descendiente).

array_pad

(PHP 4 , PHP 5)

array_pad -- Rellena una matriz con un valor hasta el tamaño especificado

Descripción

array **array_pad** (array entrada, int tama_relleno, mixed valor_relleno)

array_pad() Devuelve una copia de la *entrada* rellenada hasta el tamaño *tama_relleno* con el valor *valor_relleno*. Si *tama_relleno* es positivo, entonces la matriz es rellenada por la derecha, y si es negativo, por la izquierda. Si el valor absoluto de *tama_relleno* es menor o igual que el tamaño de la *entrada* no se produce relleno alguno.

Ejemplo 1. Ejemplo de array_pad()

```
<?php
$entrada = array (12, 10, 9);

$resultado = array_pad ($entrada, 5, 0);
// el resultado es array (12, 10, 9, 0, 0)

$resultado = array_pad ($entrada, -7, -1);
// el resultado es array (-1, -1, -1, -1, 12, 10, 9)

$resultado = array_pad ($entrada, 2, "no");
// no rellenado
?>
```

Vea también **array_fill()**, **range()**.

array_pop

(PHP 4 , PHP 5)

array_pop -- Extrae el último elemento de la matriz

Descripción

mixed **array_pop** (array &matriz)

array_pop() extrae y devuelve el último valor de la *matriz*, acortando la *matriz* en un elemento. Si *matriz* está vacía (o no es una matriz), se regresará **NULL**.

> **Nota:** Esta función ejecutará un **reset()** en el puntero **array** después de su uso.

Ejemplo 1. Ejemplo de array_pop()

```
<?php
$pila = array ("naranja", "plátano", "manzana", "frambuesa");
$fruta = array_pop ($pila);
print_r($pila);
?>
```

Después de esto, *$pila* tendrá sólo 3 elementos:

```
Array
(
    [0] => naranja
    [1] => plátano
    [2] => manzana
)
```

y *frambuesa* será asignada a *$fruta*.
Vea también: **array_push()**, **array_shift()**, y **array_unshift()**.

array_push

(PHP 4 , PHP 5)

array_push -- Inserta uno o más elementos al final de la matriz

Descripción

int **array_push** (array &matriz, mixed var [, ...])

array_push() considera a la *matriz* como una pila, e inserta las variables que se le pasan al final de la *matriz*. La longitud de la *matriz* se incrementa en el número de variables insertadas. Tiene el mismo efecto que ejecutar:

```
<?php
$array[] = $var;
?>
```

para cada *var*.

Devuelve el nuevo número de elementos de la matriz.

Ejemplo 1. Ejemplo de array_push()

```
<?php
$pila = array ("naranja", "plátano");
array_push($pila, "manzana", "frambueza");
print_r($pila);
?>
```

Este ejemplo dejará *$pila* con los siguientes elementos:

```
Array
(
    [0] => naranja
    [1] => plátano
    [2] => manzana
    [3] => frambueza
)
```

> **Nota:** Si usa **array_push()** para agregar un elemento a la matriz, es mejor usar *$array[]* = porque de esa manera no se tiene la carga de llamar a una función.

Vea también: **array_pop()**, **array_shift()**, y **array_unshift()**.

array_rand

(PHP 4 , PHP 5)

array_rand -- Selecciona una o más entradas aleatorias de una matriz

Descripción

mixed **array_rand** (array entrada [, int num_req])

array_rand() es bastante útil cuando desea elegir una o más entradas aleatorias de una matriz. Recibe una matriz de *entrada* y un argumento opcional *num_req* que especifica cuántas entradas desea seleccionar; si no se precisa éste argumento, recibe un valor por defecto de 1.

Si está eligiendo únicamente una entrada, **array_rand()** devuelve la clave de una entrada al azar. De lo contrario, devuelve una matriz de claves para las entradas aleatorias. Esto se ha hecho de esta forma para que usted pueda elegir claves al igual que valores al azar de la matriz.

> **Nota:** A partir de PHP 4.2.0, no es necesario inicializar el generador de números aleatorios con **srand()** ó **mt_srand()**, ya que esto se hace ahora automáticamente.

Ejemplo 1. Ejemplo de array_rand()

```
<?php
srand((float) microtime() * 10000000);
$entrada = array("Neo", "Morpheus", "Trinity", "Cypher", "Tank");
$claves_aleatorias = array_rand($entrada, 2);
echo $entrada[$claves_aleatorias[0]] . "\n";
echo $entrada[$claves_aleatorias[1]] . "\n";
?>
```

Vea también **shuffle()**.

array_reduce

(PHP 4 >= 4.0.5, PHP 5)

array_reduce -- Reduce iterativamente una matriz a un solo valor usando una función llamada de retorno

Descripción

mixed **array_reduce** (array entrada, callback funcion [, int inicial])

array_reduce() aplica iterativamente la función *funcion* a los elementos de la matriz *entrada*, con el propósito de reducir la matriz a un solo valor. Si el argumento opcional *inicial* está disponible, éste será usado al inicio del proceso, o como resultado final en caso de que la matriz está vacía.

Ejemplo 1. Ejemplo de array_reduce()

```
<?php
function rsum($v, $w)
{
    $v += $w;
    return $v;
}

function rmul($v, $w)
{
    $v *= $w;
    return $v;
}

$a = array(1, 2, 3, 4, 5);
$x = array();
$b = array_reduce($a, "rsum");
$c = array_reduce($a, "rmul", 10);
$d = array_reduce($x, "rsum", 1);
?>
```

Este ejemplo resultará en *$b* conteniendo el valor *15*, *$c* conteniendo *1200* (= 1*2*3*4*5*10), y *$d* conteniendo *1*.

Vea también **array_filter()**, **array_map()**, **array_unique()**, y **array_count_values()**.

array_reverse

(PHP 4 , PHP 5)

array_reverse -- Devuelve una matriz con los elementos en orden inverso

Descripción

array **array_reverse** (array matriz [, bool preserva_llaves])

array_reverse() toma la *matriz* de entrada y devuelve una nueva matriz con los elementos en orden inverso, se preservan los índices si *preserva_llaves* es **TRUE**

Ejemplo 1. Ejemplo de array_reverse()

```
<?php
$entrada = array ("php", 4.0, array ("verde", "rojo"));
$resultado = array_reverse ($entrada);
$resultado_llaves = array_reverse($entrada, true);
?>
```

Esto hace que ambos *$resultado* y *$resultado_llaves* tengan los mismos elementos, pero note la diferencia entre los índices. La salida de *$resultado* y *$resultado_llaves* será:

```
Array
(
    [0] => Array
        (
            [0] => verde
            [1] => rojo
        )

    [1] => 4
    [2] => php
)
Array
(
    [2] => Array
        (
            [0] => verde
            [1] => rojo
        )

    [1] => 4
    [0] => php
)
```

Nota: Esta función fue añadida en PHP 4.0.3.

Vea tambié no **array_flip()**.

array_search

(PHP 4 >= 4.0.5, PHP 5)

array_search -- Busca un valor determinado en una matriz y devuelve la clave correspondiente en caso de éxito

Descripción

mixed **array_search** (mixed aguja, array pajar [, bool estricto])

Busca en el *pajar* por la *aguja* y retorna la clave de ésta si se encuentra en la matriz, o **FALSE** de lo contrario.

Nota: Si *aguja* es una cadena, la comparación es realizada de forma sensible a mayúsculas y minúsculas.

> **Nota:** En versiones de PHP anteriores a 4.2.0, **array_search()** devuelve **NULL** en caso de fallo, en lugar de **FALSE**.

Si el opcional tercer argumento, *estricto*, se define como **TRUE** entonces la función **array_search()** también realizará un chequeo sobre los tipos de datos de *aguja* en el *pajar*.

Si la *aguja* es encontrada en el *pajar* más de una vez, la primera clave coincidente es devuelta. Para devolver las claves de todos los valores coincidentes, use en su lugar **array_keys()** con el parámetro opcional *valor_busqueda*.

Ejemplo 1. Ejemplo de array_search()

```
<?php
$matriz = array(0 => 'blue', 1 => 'red', 2 => 'green', 3 => 'red');

$clave = array_search('green', $matriz); // $clave = 2;
$clave = array_search('red', $matriz);   // $clave = 1;
?>
```

Aviso
Esta función puede devolver **FALSE**, pero también puede devolver un valor no-booleano que será evaluado **FALSE**, como por ejemplo *0* o "". Por favor, lea la sección Booleans para más información. Utilice el operador === para comprobar el valor devuelto por esta función.

Vea también **array_keys()**, **array_values()**, **array_key_exists()**, y **in_array()**.

array_shift

(PHP 4 , PHP 5)

array_shift -- Extrae un elemento del comienzo de la matriz

Descripción

mixed **array_shift** (array &array)

array_shift() extrae el primer valor de la *matriz* y lo devuele, acortando la *matriz* en un elemento y moviendo todo hacia abajo. Todos los índices numéricos de la matriz serán modificados para empezar desde cero, mientras que los índices literales no serán tocados. Si *array* está vacío (o no es una matriz), se regresará **NULL**

> **Nota:** Esta función ejecutará un **reset()** en el puntero **array** después de su uso.

Ejemplo 1. Ejemplo de array_shift()

```
<?php
$pila = array ("naranja", "plátano", "manzana", "frambueza");
$fruta = array_shift ($pila);
print_r($pila);
?>
```

Esto resulta en *$pila* conteniendo 3 elementos:

```
Array
(
    [0] => plátano
    [1] => manzana
    [2] => frambueza
)
```

y *naranja* es asignada a *$fruta*.

Vea también: **array_unshift()**, **array_push()**, y **array_pop()**.

array_slice

(PHP 4 , PHP 5)

array_slice -- Extrae una porción de la matriz

Descripción

array **array_slice** (array matriz, int desplazamiento [, int tamaño [, bool conserva_llaves]])

array_slice() devuelve una secuencia de elementos de la *matriz* especificada por los parámetros *desplazamiento* y *tamaño*.

Si el *desplazamiento* es positivo, la secuencia comenzará en dicha posición de la *matriz*. Si el *desplazamiento* es negativo, la secuencia comenzará en esa posición desde el final de la *matriz*.

Si se especifica el *tamaño* y éste es positivo, la secuencia contendrá tantos elementos como se diga en él. Si fuese negativo, la secuencia se detendrá a tantos elementos del final de la matriz. Si se omite, la secuencia contendrá todos los elementos desde el *desplazamiento* hasta el final de la *matriz*.

Note que **array_slice()** reasignará los índices de la matriz por defecto. Desde PHP 5.0.2, usted puede cambiar este comportamiento fijando el parámetro *conserva_llaves* a **TRUE**.

Ejemplo 1. Ejemplos de array_slice()

```
<?php
$entrada = array("a", "b", "c", "d", "e");

$salida = array_slice($input, 2);      // regresa "c", "d", and "e"
$salida = array_slice($input, -2, 1);  // regresa "d"
$salida = array_slice($input, 0, 3);   // regresa "a", "b", and "c"

// note la diferencias en las índices
print_r(array_slice($entrada, 2, -1));
print_r(array_slice($entrada, 2, -1, true));
?>
```

El resultado del ejemplo seria:

```
Array
(
    [0] => c
    [1] => d
)
Array
(
    [2] => c
    [3] => d
)
```

Vea también: **array_splice()**, **unset()**.

array_splice

(PHP 4 , PHP 5)

array_splice -- Suprime una porción de la matriz y la sustituye por otra cosa

Descripción

array **array_splice** (array &entrada, int desplazamiento [, int tamano [, array sustitucion]])

array_splice() suprime los elementos designados por el *desplazamiento* y el *tamano* de la matriz *entrada*, y los sustituye con los elementos de la matriz de *sustitucion* si se especifica.

Si el *desplazamiento* es positivo, el comienzo de la parte suprimida sería en esa posición desde el comienzo de la matriz de *entrada*. Si el *desplazamiento* es negativo, se cuenta la posición desde el final de la matriz de *entrada*.

Si se omite *tamano*, se suprime todo desde el *desplazamiento* hasta el final de la matriz. Si se especifica el *tamano* y es positivo, se suprimirán tantos elementos como se especifica. Si fuera negativo, el final de la porción eliminada estará a tantos elementos del final de la matriz. Truco: para eliminar todo desde el *desplazamiento* hasta el final de la matriz cuando también se especifica *sustitucion*, utilice *count($entrada)* como *tamano*.

Si se especifica la matriz de *sustitucion*, entonces los elementos suprimidos son reemplazados con los elementos de dicha matriz. Si los valores de *desplazamiento* y *tamano* son tales que nada es borrado, los elementos de la matriz *sustitucion* se insertarán en la posición indicada por el *desplazamiento*. Truco: si sólo se va a sustituir algo por un elemento nada más, no hace falta poner *array()* alrededor del mismo, salvo que dicho elemento sea una matriz en sí mismo.

Las siguientes funciones son ñalentes: Las sentencias siguientes cambian los valores de *$entrada* en la misma forma:

Tabla 1. ñalencias array_splice()

array_push($input, $x, $y)	array_splice($input, count($input), 0, array ($x, $y))
array_pop($input)	array_splice($input, -1)
array_shift($input)	array_splice($input, 0, 1)
array_unshift($input, $x, $y)	array_splice($input, 0, 0, array($x, $y))
$input[$x] = $y // Para matrices donde el índice ñale a la posición	array_splice($input, $x, 1, $y)

Devuelve una matriz que tiene los elementos eliminados

Ejemplo 1. Ejemplos de array_splice()

```
<?php
$entrada = array("rojo", "verde", "azul", "amarillo");

array_splice($entrada, 2);      // $entrada vale ahora array("rojo", "verde")
array_splice($entrada, 1, -1);  // $entrada vale ahora array("rojo", "amarillo")
array_splice($entrada, 1, count($entrada), "naranja");
                                // $entrada vale ahora array("rojo", "naranja")
array_splice($entrada, -1, 1, array("negro", "marr&oacute;n"));
                                // $entrada vale ahora array("rojo", "verde",
                                //          "azul", "negro", "marr&oacute;n")
?>
```

Vea también: **array_slice()**, **unset()**, **array_merge()**.

array_sum

(PHP 4 >= 4.0.4, PHP 5)

array_sum -- Calcula la suma de los valores en una matriz

Descripción

number **array_sum** (array matriz)

array_sum() devuelve la suma de los valores de una matriz como un entero o un valor de punto flotante.

Ejemplo 1. Ejemplos de array_sum()

```
<?php
$a = array(2, 4, 6, 8);
echo "sum(a) = " . array_sum($a) . "\n";

$b = array("a" => 1.2, "b" => 2.3, "c" => 3.4);
echo "sum(b) = " . array_sum($b) . "\n";
?>
```

La salida del anterior programa será:

```
sum(a) = 20
sum(b) = 6.9
```

> **Nota:** Las versiones de PHP anteriores a 4.2.1 modificaban la matriz misma pasada como argumento y convertía las cadenas a números (lo que en la mayoría de casos significaba convertirlas a cero, dependiendo de sus valores).

array_udiff_assoc

(PHP 5)

array_udiff_assoc -- Computa la diferencia entre matrices con un chequeo de índices adicional, comparando los datos con una llamada de retorno

Descripción

array **array_udiff_assoc** (array matriz1, array matriz2 [, array ..., callback func_comparacion_datos])

array_udiff_assoc() devuelve un **array** que contiene todos los valores de *matriz1* que no están presentes en ninguno de los otros argumentos. Note que las claves son usadas en la comparación, a diferencia de **array_diff()** y **array_udiff()**. La comparación entre datos de las matrices es realizada usando una llamada de retorno especificada por el usuario. En este sentido, el comportamiento de ésta función es opuesto al de **array_diff_assoc()**, la cual usa una función interna para la comparación.

Ejemplo 1. Ejemplo de array_udiff_assoc()

```php
<?php
class cr {
    private $miembro_privado;
    function cr($val)
    {
        $this->miembro_privado = $val;
    }

    function func_comp_cr($a, $b)
    {
        if ($a->miembro_privado === $b->miembro_privado) return 0;
        return ($a->miembro_privado > $b->miembro_privado)? 1:-1;
    }
}

$a = array("0.1" => new cr(9), "0.5" => new cr(12), 0 => new cr(23), 1=> new cr(4), 2 =
$b = array("0.2" => new cr(9), "0.5" => new cr(22), 0 => new cr(3), 1=> new cr(4), 2 =>

$resultado = array_udiff_assoc($a, $b, array("cr", "func_comp_cr"));
print_r($resultado);
?>
```

El resultado es:

```
Array
(
    [0.1] => cr Object
        (
            [miembro_privado:private] => 9
        )

    [0.5] => cr Object
        (
            [miembro_privado:private] => 12
        )

    [0] => cr Object
        (
            [miembro_privado:private] => 23
        )
)
```

En nuestro ejemplo anterior puede observar la pareja *"1" => new cr(4)* presente en ambas matrices y por lo tanto no hace parte de la salida de la función.

Para la comparación es usada la llamada de retorno entregada por el usuario. Ésta debe devolver un entero menor que, igual, o mayor que cero si el primer argumento es considerado como menor, igual, o mayor que el segundo, respectivamente.

> **Nota:** Por favor note que esta función sólo analiza una dimensión de una matriz n-dimensional. Por supuesto, puede analizar dimensiones más profundas usando, por ejemplo, *array_udiff_assoc($matriz1[0], $matriz2[0], "func_alguna_comparacion");*.

Vea también **array_diff()**, **array_diff_assoc()**, **array_diff_uassoc()**, **array_udiff()**, **array_udiff_uassoc()**, **array_intersect()**, **array_intersect_assoc()**, **array_uintersect()**, **array_uintersect_assoc()** y **array_uintersect_uassoc()**.

array_udiff_uassoc

(PHP 5)

array_udiff_uassoc -- Computa la diferencia entre matrices con un chequeo de índices adicional, comparando los datos y los índices con una llamada de retorno

Descripción

array **array_udiff_uassoc** (array matriz1, array matriz2 [, array ..., callback func_comparacion_datos, callback func_comparacion_claves])

array_udiff_uassoc() devuelve un **array** que contiene todos los valores de *matriz1* que no están presentes en ninguno de los otros argumentos. Note que las claves son usadas en la comparación, a diferencia de **array_diff()** y **array_udiff()**. La comparación de los datos de las matrices es realizada usando una llamada de retorno entregada por el usuario: *func_comparacion_datos*. En este sentido, su comportamiento es el opuesto del de **array_diff_assoc()**, quien usa una función interna para la comparación. La comparación de claves (índices) es realizada también por la llamada de retorno *func_comparacion_claves*. Este comportamiento contrasta con lo que hace **array_udiff_assoc()**, ya que ésta compara los índices usando una función interna.

Ejemplo 1. Ejemplo de array_udiff_uassoc()

```
<?php
class cr {
    private $miembro_privado;
    function cr($val)
    {
        $this->miembro_privado = $val;
    }

    function func_comp_cr($a, $b)
    {
        if ($a->miembro_privado === $b->miembro_privado) return 0;
        return ($a->miembro_privado > $b->miembro_privado)? 1:-1;
    }

    function func_comp_claves($a, $b)
    {
        if ($a === $b) return 0;
        return ($a > $b)? 1:-1;
    }
}
$a = array("0.1" => new cr(9), "0.5" => new cr(12), 0 => new cr(23), 1=> new cr(4), 2 =
$b = array("0.2" => new cr(9), "0.5" => new cr(22), 0 => new cr(3), 1=> new cr(4), 2 =>

$resultado = array_udiff_uassoc($a, $b, array("cr", "func_comp_cr"), array("cr", "func_
print_r($resultado);
?>
```

El resultado es:

```
Array
(
    [0.1] => cr Object
        (
            [miembro_privado:private] => 9
        )

    [0.5] => cr Object
        (
            [miembro_privado:private] => 12
        )

    [0] => cr Object
        (
            [miembro_privado:private] => 23
        )
)
```

En nuestro ejemplo anterior, puede ver que la pareja *"1" => new cr(4)* está presente en ambas matrices, y por lo tanto no hace parte de la salida de la función. Tenga en cuenta que debe especificar 2 llamadas de retorno.

Para la comparación, se usa la llamada de retorno indicada por el usuario. Ésta debe devolver un entero menor que, igual, o mayor que cero si el primer argumento es considerado como menor, igual, o mayor que el segundo, respectivamente.

> **Nota:** Por favor note que esta función únicamente chequea una dimensión de una matriz n-dimensional. Por supuesto, puede chequear dimensiones más profundas usando, por ejemplo, *array_udiff_uassoc($matriz1[0], $matriz2[0], "func_comparacion_datos", "func_comparacion_claves");*.

Vea también **array_diff()**, **array_diff_assoc()**, **array_diff_uassoc()**, **array_udiff()**, **array_udiff_assoc()**, **array_intersect()**, **array_intersect_assoc()**, **array_uintersect()**, **array_uintersect_assoc()** y **array_uintersect_uassoc()**.

array_udiff

(PHP 5)

array_udiff -- Computa la diferencia entre matrices, usando una llamada de retorno para la comparación de datos

Descripción

array **array_udiff** (array matriz1, array matriz2 [, array ..., callback func_comparacion_datos])

array_udiff() devuelve una matriz que contiene todos los valores de *matriz1* que no están presentes en ninguno de los otros argumentos. Note que las claves son preservadas. Para la comparación de datos, se usa *func_comparacion_datos*. Esta función debe devolver un entero menor que, igual, o mayor que cero si el primer argumento es considerado como menor, igual, o mayor que el segundo, respectivamente. Esto en contraste a **array_diff()**, la cual usa una función interna para la comparación de datos.

Ejemplo 1. Ejemplo de array_udiff()

```
<?php
class cr {
    private $miembro_privado;
    function cr($val)
    {
        $this->miembro_privado = $val;
    }

    function func_comp_cr($a, $b)
    {
        if ($a->miembro_privado === $b->miembro_privado) return 0;
        return ($a->miembro_privado > $b->miembro_privado)? 1:-1;
    }
}
$a = array("0.1" => new cr(9), "0.5" => new cr(12), 0 => new cr(23), 1=> new cr(4), 2 =
$b = array("0.2" => new cr(9), "0.5" => new cr(22), 0 => new cr(3), 1=> new cr(4), 2 =>

$resultado = array_udiff($a, $b, array("cr", "func_comp_cr"));
print_r($resultado);
?>
```

El resultado es:

```
Array
(
    [0.5] => cr Object
        (
            [miembro_privado:private] => 12
        )

    [0] => cr Object
        (
            [miembro_privado:private] => 23
        )

)
```

Nota: Dos elementos son considerados iguales si, y solo si *(string) $elem1 === (string) $elem2*. Esto es: cuando su representación tipo cadena es la misma.

Nota: Por favor note que esta función sólo analiza una dimensión de una matriz n-dimensional. Por supuesto, puede analizar dimensiones más profundas usando *array_udiff($matriz1[0], $matriz2[0], "data_compare_func");*.

Vea también **array_diff()**, **array_diff_assoc()**, **array_diff_uassoc()**, **array_udiff_assoc()**, **array_udiff_uassoc()**, **array_intersect()**, **array_intersect_assoc()**, **array_uintersect()**, **array_uintersect_assoc()** y **array_uintersect_uassoc()**.

array_uintersect_assoc

(PHP 5)

array_uintersect_assoc -- Calcula la intersección de matrices con chequeo adicional de índices, comparando los datos por una función del usuario

Descripción

array **array_uintersect_assoc** (array array1, array array2 [, array ..., callback data_compare_func])

array_uintersect_assoc() regresa una matriz conteniendo todos los valores de *array1* que están presentes en todos los argumentos. Note que los índices son usados en la comparación a diferencia de **array_uintersect()**. Los datos son comparados usando una función del usuario.

Ejemplo 1. Ejemplo array_uintersect_assoc()

```
<?php
$array1 = array("a" => "green", "b" => "brown", "c" => "blue", "red");
$array2 = array("a" => "GREEN", "B" => "brown", "yellow", "red");

print_r(array_uintersect_assoc($array1, $array2, "strcasecmp"));
?>
```

El resultado del ejemplo seria:

```
Array
(
    [a] => green
)
```

Para la comparación es usada la función dada por el usuario. La cual debe regresar un entero menor que, igual a, o mayor que cero si el primer argumento es considerado ser, respectivamente, menor que, igual a, o mayor que el segundo argumento.

Vea también **array_uintersect()**, **array_intersect_assoc()**, **array_intersect_uassoc()** y

array_uintersect_uassoc().

array_uintersect_uassoc

(PHP 5)

array_uintersect_uassoc -- Calcula la intersección de matrices con chequeo adicional de índices, compara los datos y los índices por una función del usuario

Descripción

array **array_uintersect_uassoc** (array array1, array array2 [, array ..., callback data_compare_func, callback key_compare_func])

array_uintersect_uassoc() regresa una matriz que contiene todos los valores de *array1* que están presentes en todos los argumentos. Note que los índices son usados en la comparación a diferencia de **array_uintersect()**. Los datos y los índices son comparados usando una función del usuario.

Ejemplo 1. Ejemplo array_uintersect_uassoc()

```
<?php
$array1 = array("a" => "green", "b" => "brown", "c" => "blue", "red");
$array2 = array("a" => "GREEN", "B" => "brown", "yellow", "red");

print_r(array_uintersect_uassoc($array1, $array2, "strcasecmp", "strcasecmp"));
?>
```

El resultado del ejemplo seria:

```
Array
(
    [a] => green
    [b] => brown
)
```

Para la comparación es usada una función dada por el usuario. La cual debe regresar un entero menor que, igual a, o mayor que cero si el primer argumento es considerado ser menor que, igual a o mayor que el segundo argumento respectivamente.

Vea también **array_uintersect()**, **array_intersect_assoc()**, **array_intersect_uassoc()** y **array_uintersect_assoc()**.

array_uintersect

(PHP 5)

array_uintersect -- Calcula la intersección de matrices, compara los datos con una función del usuario

Descripción

array **array_uintersect** (array array1, array array2 [, array ..., callback data_compare_func])

array_uintersect() regresa una matriz que contiene todos los valores de *array1* que están presentes en topos los argumentos. Los datos son comparados usando una función del usuario.

Ejemplo 1. Ejemplo array_uintersect()

```
<?php
$array1 = array("a" => "green", "b" => "brown", "c" => "blue", "red");
$array2 = array("a" => "GREEN", "B" => "brown", "yellow", "red");

print_r(array_uintersect($array1, $array2, "strcasecmp"));
?>
```

El resultado del ejemplo seria:

```
Array
(
    [a] => green
    [b] => brown
    [0] => red
)
```

Para la comparació es usada una función del usuario. La cual debe regresar un entero menor que, igual a, o mayor que cerp si el primer argumento es considerado ser menor que, igual a, o mayor que el segundo argumento, respectivamente.

Vea también **array_intersect()**, **array_uintersect_assoc()**, **array_intersect_uassoc()** y **array_uintersect_uassoc()**.

array_unique

(PHP 4 >= 4.0.1, PHP 5)

array_unique -- Remueve valores duplicados de una matriz

Descripción

array **array_unique** (array matriz)

array_unique() toma la *matriz* de entrada y devuelve una nueva matriz sin los valores repetidos.

Note que las claves son preservadas. **array_unique()** ordena los valores tratados como cadenas inicialmente, y luego conservará la primera clave encontrada para cada valor, ignorando todas las claves posteriores. No quiere decir esto que la clave del primer valor relacionado de la *matriz* no-ordenada se conservará.

> **Nota:** Dos elementos son considerados ñalentes si y solo si *(string) $elem1 === (string) $elem2*. En palabras: cuando la representación tipo cadena es la misma.
>
> Se usará el primer elemento.

Ejemplo 1. Ejemplo de array_unique()

```
<?php
$entrada = array("a" => "verde", "rojo", "b" => "verde", "azul", "rojo");
$resultado = array_unique($entrada);
print_r($resultado);
?>
```

Esto producirá la salida:

```
Array
(
    [a] => verde
    [0] => rojo
    [1] => azul
)
```

Ejemplo 2. array_unique() y tipos de datos

```
<?php
$entrada = array(4, "4", "3", 4, 3, "3");
$resultado = array_unique($entrada);
var_dump($resultado);
?>
```

Este script resultará en la siguiente salida:

```
array(2) {
  [0] => int(4)
  [2] => string(1) "3"
}
```

array_unshift

(PHP 4 , PHP 5)

array_unshift -- Introduce uno o más elementos al principio de la matriz

Descripción

int **array_unshift** (array &matriz, mixed var [, mixed ...])

array_unshift() añade los elementos que se le pasan al frente de la *matriz*. Nótese que la lista de elementos es añadida como un todo, de modo que los elementos añadidos mantienen su orden. Todos los ídices numéricos de la matriz serán modificados para iniciar a contar desde cero mientras que los índices alfanuméricos no serán modificados.

Devuelve el número de elementos en la *matriz*.

Ejemplo 1. Ejemplo de array_unshift()

```
<?php
$queue = array("orange", "banana");
array_unshift($queue, "apple", "raspberry");
?>
```

Esto resultará que *$queue* tenga los siguientes elementos:

```
Array
(
    [0] => apple
    [1] => raspberry
    [2] => orange
    [3] => banana
)
```

Vea tambinén **array_shift()**, **array_push()**, and **array_pop()**.

array_values

(PHP 4 , PHP 5)

array_values -- Devuelve todos los valores de una matriz

Descripción

array **array_values** (array entrada)

array_values() devuelve todos los valores de la matriz *entrada* en orden numérico.

Ejemplo 1. Ejemplo de array_values()

```
<?php
$matriz = array("talla" => "XL", "color" => "dorado");
print_r(array_values($matriz));
?>
```

El resultado del ejemplo seria:

```
Array
(
    [0] => XL
    [1] => dorado
)
```

Vea también **array_keys()**.

array_walk_recursive

(PHP 5)

array_walk_recursive -- Aplicar una función de usuario recursivamente a cada miembro de una matriz

Descripción

bool **array_walk_recursive** (array &entrada, callback nombre_func [, mixed datos_usuario])

Aplica la función definida por el usuario *nombre_func* a cada elemento de la matriz *entrada*. Esta función opera de forma recursiva sobre matrices profundas. Por lo general, *nombre_func* recibe dos parámetros. El valor del parámetro *entrada* es el primero, y la clave/índice es el segundo. Si se define el parámetro opcional *datos_usuario*, éste será pasado como el tercer parámetro a la llamada de retorno *nombre_func*.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

> **Nota:** Si *nombre_func* necesita trabajar con los valores reales de la matriz, especifique el primer parámetro de *nombre_func* como una referencia. Entonces, cualquier cambio realizado sobre esos elementos será efectuado sobre la matriz original misma.

Ejemplo 1. Ejemplo de array_walk_recursive()

```
<?php
$dulce = array('a' => 'manzana', 'b' => 'banano');
$frutas = array('dulce' => $dulce, 'acido' => 'limon');

function prueba_imprimir($item, $clave)
{
    echo "$clave contiene $item\n";
}

array_walk_recursive($frutas, 'prueba_imprimir');
?>
```

La salida del anterior programa será:

```
a contiene manzana
b contiene banano
acido contiene limon
```

Notará que la clave 'dulce' nunca es mostrada. Cualquier clave que contenga un valor **array** no será pasada a la función.

Vea también **array_walk()**.

array_walk

(PHP 3>= 3.0.3, PHP 4 , PHP 5)

array_walk -- Aplica una función del usuario a cada elemento de una matriz.

Descripción

int **array_walk** (array &matriz, string func [, mixed datosvarios])

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Aplica la función llamada *func* a cada elemento de la *matriz*. La función *func* recibirá el valor de la matriz como primer parámetro y la clave como segundo. Si se proporciona el parámetro *datosvarios* será pasado como tercer parámetro a la función de usuario.

Si *func* necesita más de dos o 3 argumentos, dependiendo de *datosvarios*, se generará un aviso cada vez que **array_walk()** llama a *func*. Estos avisos pueden suprimirse si se pone @ antes de la llamada a **array_walk()**, o usando la función **error_reporting()**.

> **Nota:** Si *func* precisa trabajar con los valores reales de la matriz, especifique que el valor del primer parámetro de *func* debe pasarse por referencia. Desde ese instante, los cambios realizados sobre dichos elementos también serán realizados en la propia matriz.
>
> **Nota:** El pasar la clave y los datos de usuario a *func* fue una característica añadida en PHP 4.0.0

array_walk() no es afectado por el apuntador interno del parámetro *matriz*. **array_walk()** avanzará por toda la matriz sin importar la posición del apuntador. Para reinicializar el apuntador, use **reset()**. en PHP 3, **array_walk()** reinicializa el apuntador.

No se debe cambiar la matriz desde la llamada a la función, ej. agregar/borrar elementos, vaciar elementos, etc. si la matriz que está siendo usada por **array_walk()** cambia, el comportamiento de esta función será indefinido e impredecible.

Ejemplo 1. Ejemplo de array_walk()

```
<?php
$frutas = array ("d"=>"limon", "a"=>"naranja", "b"=>"platano", "c"=>"manzana");

function test_alterar ($item1, $clave, $prefix) {
   $item1 = "$prefix: $item1";
}

function test_ver ($item2, $clave) {
   echo "$clave. $item2<br />\n";
}

echo "Antes...:\n";
array_walk ($frutas, 'test_ver');

array_walk ($frutas, 'test_alterar', 'fruta');
echo "...y despues:\n";

array_walk ($frutas, 'test_ver');
?>
```

El resultado del ejemplo seria:

```
Antes...:
d. limon
a. naranja
b. platano
c. manzana
...y despues:
d. fruta: limon
a. fruta: naranja
b. fruta: platano
c. fruta: manzana
```

Vea también **array_walk_recursive()**, **create_function()**, **list()**, foreach, **each()**, **call_user_func_array()**, y **array_map()**.

array

(PHP 3, PHP 4, PHP 5)

array -- Crear una matriz

Descripción

array **array** (mixed ...)

Devuelve una matriz con los parámetros que se le pasan. A dichos parámetros se les puede dar un índice usando el operador =>. Lea la sección sobre los tipos de matrices para más información sobre matrices.

> **Nota: array()** es una construcción del lenguaje que se utiliza para representar matrices literales, no una función regular.

La forma "índice => valor" separada por comas, define índices y valores. el índice puede ser de tipo cadena o numérico. Cuando el índice es omitido, se genera automáticamente un índice numérico, empezando en cero. Si el índice es un entero, el siguiente índice generado será igual al índice con número mayor + 1. Note que cuando se definen dos índices idénticos, el último sobre escribe al primero.

Tener una coma después del último elemento de la matriz, aunque inusual, es sintácticamente valido.

El siguiente ejemplo demuestra cómo crear una matriz bidimensional, cómo especificar claves para matrices asociativas, y cómo especificar índices no consecutivos en matrices normales.

Ejemplo 1. Ejemplo de array()

```
<?php
$frutas = array (
    "frutas"  => array("a"=>"naranja", "b"=>"pl&aacute;tano", "c"=>"manzana"),
    "n&uacute;meros" => array(1, 2, 3, 4, 5, 6),
    "hoyos"   => array("primero", 5 => "segundo", "tercero")
);
?>
```

Ejemplo 2. Índice automático con array()

```
<?php
$array = array(1, 1, 1, 1,  1, 8 => 1,  4 => 1, 19, 3 => 13);
print_r($array);
?>
```

El resultado del ejemplo seria:

```
Array
(
    [0] => 1
    [1] => 1
    [2] => 1
    [3] => 13
    [4] => 1
    [8] => 1
    [9] => 19
)
```

Note que el índice '3' es definido dos veces, y permanece su valor final de '13'. El índice 4 es definido depués del índice 8, y en seguida se genera el índice 9 (para el valor 19), porque el índice mayor era 8.

Este ejemplo crea una matriz en donde los índices inician en 1.

Ejemplo 3. Índice base 1 con array()

```
<?php
$primercuarto = array(1 => 'Enero', 'Febrero', 'Marzo');
print_r($primercuarto);
?>
```

El resultado del ejemplo seria:

```
Array
(
    [1] => Enero
    [2] => Febrero
    [3] => Marzo
)
```

Como en Perl, puede accesar un valor de la matriz desde dentro de una cadena contenida en dobles comillas. Sin embargo, con PHP necesitará encerrar la matriz entre las llaves curvas.

Ejemplo 4. Accesando una matriz dentro de una cadena

```
<?php

$foo = array('bar' => 'baz');
echo "Hello {$foo['bar']}!"; // Hello baz!

?>
```

Vea también **array_pad()**, **list()**, **count()**, foreach, **range()**.

arsort

(PHP 3, PHP 4 , PHP 5)

arsort -- Ordena una matriz en orden inverso y mantiene la asociación de índices

Descripción

bool **arsort** (array &matriz [, int sort_flags])

Esta función ordena una matriz de modo que los índices mantengan su correlación con los elementos de la misma a los que están asociados. Esto se utiliza principalmente para ordenar matrices asociativas en las que el orden de los elementos es importante.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplo 1. Ejemplo de arsort()

```
<?php
$frutas = array ("d"=>"limon", "a"=>"naranja", "b"=>"platano", "c"=>"manzana");
arsort ($frutas);
reset(&futas);
while (list($key, $val = each($frutas)) {
   echo "$key = $val\n";
}
?>
```

El resultado del ejemplo seria:

```
b = platano
a = naranja
c = manzana
d = limon
```

Las frutas han sido ordenadas en orden alfabético inverso y los índices asociados con cada elemento se han mantenido.

Puede modificar el comportamiento de orden usando el parámetro opcional *sort_flags* para detalles vea **sort()**.

Vea también: **asort()**, **rsort()**, **ksort()**, y **sort()**.

asort

(PHP 3, PHP 4 , PHP 5)

asort -- Ordena una matriz y mantiene la asociación de índices

Descripción

bool **asort** (array &matriz [, int sort_flags])

Esta función ordena una matriz de modo que los índices mantengan su correlación con los elementos de la misma a los que están asociados. Esto se utiliza principalmente para ordenar matrices asociativas en las que el orden de los elementos es importante.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplo 1. Ejemplo de asort()

```
>?php
$frutas = array ("d"=>"limon", "a"=>"naranja", "b"=>"platano", "c"=>"manzana");
asort($frutas);
reset($frutas);
while (list($key, $val) = each($frutas)) {
   echo "$key = $val\n";
}
?>
```

El resultado del ejemplo seria:

```
d = limon
a = naranja
c = manzana
d = platano
```

Las frutas han sido ordenadas en orden alfabético y los índices asociados con cada elemento se han mantenido.

Puede modificar el comportamiento de orden usando el parámetro opcional *sort_flags* para detalles vea **sort()**.

Vea también: **arsort()**, **rsort()**, **ksort()**, y **sort()**.

compact

(PHP 4 , PHP 5)

compact -- Crea una matriz que contiene variables y sus valores

Descripción

array **compact** (mixed varname [, mixed ...])

compact() toma un número variable de parámetros. Cada uno puede ser tanto una cadena que contiene el nombre de la variable, como una matriz de nombres de variable. La matriz puede contener otras matrices de nombres de variable en su interior; **compact()** los procesa recursivamente.

Para cada uno de estos, **compact()** busca una variable con dicho nombre en la tabla de símbolos y la añade a la matriz de salida de modo que el nombre de la variable es la clave y el contenido de ésta es el valor para dicha clave. Para resumir, hace lo contrario de **extract()**. Devuelve la matriz de salida con las variables añadidas a la misma.

Cualquier cadena que no haya sido definida simplemente se evitará.

> **Gotcha:** A causa de que Variables variables no puede ser usada con las Superglobal arrays de PHP dentro de funciones, las matrices Superglobal no pueden ser pasadas a **compact()**.

Ejemplo 1. Ejemplo de compact()

```php
<?php
$ciudad = "San Francisco";
$estado = "CA";
$evento = "SIGGRAPH";

$location_vars = array ("ciudad", "estado");

$resultado = compact ("evento", "nada_aqui", $location_vars);
?>
```

Tras esto, *$resultado*

```
Array
(
    [event] => SIGGRAPH
    [city] => San Francisco
    [state] => CA
)
```

Vea también: **extract()**.

count

(PHP 3, PHP 4 , PHP 5)

count -- Cuenta los elementos de una matriz o propiedades de un objeto

Descripción

int **count** (mixed var [, int mode])

Devuelve el número de elementos en *var*, que típicamente es una matriz, porque cualquier otra cosa diferente de un objeto tendría sólo un elemento.

Para objetos **count()** regresará el número de propiedades no estáticas, sin tomar en cuenta su visibilidad. Si tiene instalado SPL, enlazarlo a **count()** implementando la interface *Countable*. La interface tiene exactamente un método, **count()**, el cual regresa el mismo valor que regresarí la función **count()**.

Si *var* no es una matriz o un objeto, se regresará *1*. Hay una excepción, si *var* es **NULL**, se regresará *0*.

> **Nota:** El parámetro opcional *mode* está disponible desde PHP 4.2.0.

Si el parámetro opcional *mode* es iniciado a **COUNT_RECURSIVE** (o 1), **count()** contará recursivamente la matriz. Esto es útil particularmente para contar todos los elementos de una matriz multidimensional. El valor por defecto para *mode* es *0*. **count()** no detecta recursión infinita.

Atención
count() puede regresar 0 para una variable que no ha sido inicializada, pero también regresa 0 para una variable que ha sido inicializada con una matriz vacía. Use **isset()** para probar si la variable ha sido definida.

Por favor vea la sección del manual sobre Array para una explicación más detallada de como son usadas e implementadas las matrices en PHP.

Ejemplo 1. Ejemplo

```php
<?php
$a[0] = 1;
$a[1] = 3;
$a[2] = 5;
$result = count($a);
// $result == 3

$b[0]  = 7;
$b[5]  = 9;
$b[10] = 11;
$result = count($b);
// $result == 3;

$result = count(null);
// $result == 0;
$result = count(false);
// $result == 1;

$obj = new StdClass;
$obj->foo = 'A property';
$obj->bar = 'Another property';
$result = count($obj);
// $result == 2;

?>
```

Ejemplo 2. Ejemplo recursivo (PHP >= 4.2.0)

```php
<?php
$food = array('fruits' => array('orange', 'banana', 'apple'),
              'veggie' => array('carrot', 'collard', 'pea'));

// recursive count
echo count($food, COUNT_RECURSIVE);  // output 8

// normal count
echo count($food);                   // output 2

?>
```

Vea también **is_array()**, **isset()**, and **strlen()**.

current

(PHP 3, PHP 4 , PHP 5)

current -- Devuelve el elemento actual de una matriz

Descripción

mixed **current** (array &matriz)

Cada matriz tiene un puntero interno al elemento "actual", que se inicializa al primer elemento insertado en la misma.

La función **current()** simplemente devuelve el elemento de la tabla al que apunta el puntero interno. No mueve el puntero de ninguna manera. Si el puntero interno apunta fuera del final de la lista de elementos, **current()** devuelve **`FALSE`**.

Aviso

Si la matriz contiene elementos vacíos (0 o "", la cadena vacía) esta función devolverá **FALSE** también para dichos elementos. Esto hace imposible determinar si se está realmente al final de la lista en tales matrices usando **current()**. Para recorrer adecuadamente una matriz que pueda contener elementos vacíos, utilice la función **each()**.

Ejemplo 1. Ejemplo: uso de current() y amigos

```
<?php
$transport = array('foot', 'bike', 'car', 'plane');
$mode = current($transport); // $mode = 'foot';
$mode = next($transport);    // $mode = 'bike';
$mode = current($transport); // $mode = 'bike';
$mode = prev($transport);    // $mode = 'foot';
$mode = end($transport);     // $mode = 'plane';
$mode = current($transport); // $mode = 'plane';
?>
```

Vea también: **end()**, **key()**, **next()**, **prev()** y **reset()**.

each

(PHP 3, PHP 4 , PHP 5)

each -- Devuelve el siguiente par clave/valor de una matriz y avanza el apuntador

Descripción

array **each** (array &matriz)

Devuelve el par clave/valor actual para la *matriz* y avanza el puntero de la misma. Esta pareja se devuele en una matriz de 4 elementos, con las claves *0*, *1*, *key*, y *value*. Los elementos *0* y *key* contienen el nombre de clave del elemento de la matriz, y *1* y *value* contienen los datos.

Si el puntero interno para la matriz apunta después del final del contenido de la matriz, **each()** devuelve **FALSE**.

Ejemplo 1. Ejemplos de each()

```
<?php
$chorrada = array ("bob", "fred", "jussi", "jouni", "egon", "marliese");
$tonteria = each ($chorrada);
print_r($tonteria);
?>
```

$tonteria ahora contiene los siguientes pares de llave/valor:

```
Array
(
    [1] => bob
    [value] => bob
    [0] => 0
    [key] => 0
)
```

```
<?php
$foo = array("Robert" => "Bob", "Seppo" => "Sepi");
$bar = each($foo);
print_r($bar);
?>
```

$tonteria ahora contiene los siguientes pares de llave/valor:

```
Array
(
    [1] => Bob
    [value] => Bob
    [0] => Robert
    [key] => Robert
)
```

each() se usa normalmente de forma conjunta a **list()** para recorrer una matriz; por ejemplo:

Ejemplo 2. Recorriendo una matriz con each()

```
<?php
$fruit = array('a' => 'apple', 'b' => 'banana', 'c' => 'cranberry');

reset($fruit);
while (list($key, $val) = each($fruit)) {
    echo "$key => $val\n";
}
?>
```

El resultado del ejemplo seria:

```
a => apple
b => banana
c => cranberry
```

Cuando se ha ejecutado **each()**, el cursor de la matriz quedará en el siguiente elemento de la misma, o en el último si llega al final de ésta. Tiene que usar **reset()** si quiere recorrer la matriz otra vez usando each.

Atención
A causa de que asignar una matriz a otra variable reinicia el apuntador original de la matriz, nuestro ejemplo anterior pudo causar un ciclo sin fin tuvimos que asignar *$fruit* a otra variable dentro del ciclo.

Vea también: **key()**, **list()**, **current()**, **reset()**, **next()**, **prev()**, y foreach.

end

(PHP 3, PHP 4 , PHP 5)

end -- Mueve el puntero interno de una tabla al último elemento

Descripción

mixed **end** (array &matriz)

end() avanza el puntero interno de la *matriz* al último elemento, y regresa su valor.

Ejemplo 1. Un Ejemplo simple de end()

```
<?php

$fruits = array('apple', 'banana', 'cranberry');
echo end($fruits); // cranberry

?>
```

Vea también: **current()**, **each()**, **prev()**, **next()**, y **reset()**.

extract

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

extract -- Importa variables a la tabla de símbolos desde una matriz

Descripción

void **extract** (array matriz_vars [, int tipo_extraccion [, string prefijo]])

Esta función se utiliza para importar variables desde una matriz a la tabla de símbolos actual. Toma la matriz asoiativa *matriz_vars* y trata las claves como nombres de variable y los valores como los valores de éstas. Para cada par clave/valor creará una variable en la tabla de símbolos actual, sujeto a los parámetros *tipo_extraccion* y *prefijo*.

> **Nota:** Empezando de la versión 4.0.5, esta función regresa el número de variables extraídas.
>
> **Nota:** **`EXTR_IF_EXISTS`** y **`EXTR_PREFIX_IF_EXISTS`** fueron introducidas en la versión 4.2.0.
>
> **Nota:** **`EXTR_REFS`** fue introducida en la versión 4.3.0.

extract() controla las colisiones con las variables que ya existen. La forma de tratar éstas se determina por el *tipo_extraccion*. Puede tener únicamente uno de los siguientes valores:

EXTR_OVERWRITE

> Si hay colisión, sobreescribe la variable existente.

EXTR_SKIP

> Si hay colisión, no sobreescribas la variable existente.

EXTR_PREFIX_SAME

> Si hay una colisión, añade el *prefijo* a la nueva variable.

EXTR_PREFIX_ALL

> Añade el *prefijo* a todas las variables.

`EXTR_PREFIX_INVALID`

> Solo agrega el *prefijo* a nombres de variables invalidas/numéricas. Este fue agregado en PHP 4.0.5.

`EXTR_IF_EXISTS`

> Solo sobreescribe la variable si ya existe en la tabla de símbolos actual, de otra manera no hace nada. Esto es útil para definir una lista de variables validas y entonces extraer solo aquellas variables que esten definidas fuera de $_REQUEST por ejemplo. Esta bandera fue

agregada en PHP 4.2.0.

EXTR_PREFIX_IF_EXISTS

Solo crea los nombres de variables con el prefijo si la versión de la variable sin prefijo existe en la tabla de símbolos. Esta bandera fue agregada en PHP 4.2.0.

EXTR_REFS

Extrae las variables como referencias. Esto efectivamente significa que los valores de las variables importadas están aún referenciando a los valores del parámetro *matriz_var*. Puede usar esta bandera por sí sola o combinada con cualquier otra bandera haciendo (OR) el parámetro *tipo_extraccion*. Esta bandera fue agregada en PHP 4.3.0.

Si no se especifica *tipo_extraccion*, se asume que vale EXTR_OVERWRITE.

Nótese que el *prefijo* sólo se necesita si *tipo_extraccion* vale **EXTR_PREFIX_SAME**, **EXTR_PREFIX_ALL**, **EXTR_PREFIX_INVALID** o **EXTR_PREFIX_IF_EXISTS**. Si el resultado con prefijo no es un nombre de variable valido, no es importado en la tabla de símbolos.

extract() regresa el número de variables exitosamente importadas en la tabla de símbolos.

Aviso
No use **extract()** en datos no confiables, como entradas de usuario ($_GET, ...). pero si lo hace, por ejemplo, si quiere correr codigo anterior que confía temporalmente en register_globals, asegurese de que usa una de los valores de no-sobreescribir del parámetro *tipo_extraccion* tales como **EXTR_SKIP** y asegurese de extrar las variables *$_SERVER*, *$_SESSION*, *$_COOKIE*, *$_POST* y *$_GET* ese orden.

Un uso posible para extract sería importar en la tabla de símbolos las variables contenidas en la matriz asociativa que devuelve **wddx_deserialize()**.

Ejemplo 1. Ejemplo de extract()

```
<?php

/* Suponemos que $matriz_var es una matriz devuelta por
   wddx_deserialize */

$tamano = "grande";
$matriz_var = array ("color" => "azul",
                     "tamano"  => "media",
                     "forma" => "esfera");
extract ($matriz_var, EXTR_PREFIX_SAME, "wddx");

print "$color, $tamano, $forma, $wddx_tamano\n";

?>
```

El resultado del ejemplo seria:

```
azul, grande, esfera, media
```

La variable *$tamano* no fue sobreescrita porque especificamos **EXTR_PREFIX_SAME**, que provocó la creación de *$wddx_tamano*. Si se hubiera especificado **EXTR_SKIP**, *$wddx_tamano* ni siquiera habría sido creada. **EXTR_OVERWRITE** habría provocado que *$tamano* tuviera el valor "media", y **EXTR_PREFIX_ALL** habría provocado que aparecieran nuevas variables llamadas *$wddx_color*, *$wddx_tamano*, y *$wddx_forma*.

Debe usar matrices asociativas, las matrices numéricamente indexadas no producirán resultados a

menos que use **EXTR_PREFIX_ALL** o **EXTR_PREFIX_INVALID**.

Vea también **compact()**.

in_array

(PHP 4 , PHP 5)

in_array -- Revisa si un valor existe en una matriz

Descripción

bool **in_array** (mixed aguja, array pajar [, bool strict])

Busca la *aguja* en el *pajar*, y devuelve **TRUE** si se encuentra y **FALSE** en caso contrario.

Si el tercer parámetro *strict* es definido en **TRUE** entonces la función **in_array()** también revisará los tipos de la *aguja* en el *pajar*.

> **Nota:** Si *needle* es una cadena, la comparación es hecha de una forma sensible a mayúsculas y minúsculas.
>
> **Nota:** En PHP antes de 4.2.0. *needle* no le era permitido ser una matriz.

Ejemplo 1. Ejemplo de in_array()

```
<?php
$os = array ("Mac", "NT", "Irix", "Linux");
if (in_array ("Irix", $os))
    print "Encontrado Irix";
if (in_array("mac", $os)) {
    echo "Encontrado mac";
}
?>
```

La segunda condición falla porque **in_array()** es CASE-SENSITIVE, por lo que el programa desplegará:

```
Got Irix
```

Ejemplo 2. Ejemplo de STRICT in_array()

```
<?php
$a = array('1.10', 12.4, 1.13);

if (in_array('12.4', $a, true)) {
    echo "'12.4' Encontrado con chequeo STRICT\n";
}

if (in_array(1.13, $a, true)) {
    echo "1.13 Encontrado con chequeo STRICT\n";
}
?>
```

El resultado del ejemplo seria:

```
1.13 Encontrado con chequeo STRICT
```

Ejemplo 3. in_array() with an array as needle

```
<?php
$a = array(array('p', 'h'), array('p', 'r'), 'o');

if (in_array(array('p', 'h'), $a)) {
    echo "Se encontro 'ph'\n";
}

if (in_array(array('f', 'i'), $a)) {
    echo "Se encontro 'fi'\n";
}

if (in_array('o', $a)) {
    echo "Se encontro 'o'\n";
}
?>
```

El resultado del ejemplo seria:

```
Se encontro 'ph'
  Se encontro 'o'
```

Vea también **array_search()**, **array_key_exists()**, y **isset()**.

key

(PHP 3, PHP 4 , PHP 5)

key -- Obtiene una clave de una matriz asociativa

Descripción

mixed **key** (array &key)

key() devuelve el elemento índice de la posición actual en la matriz.

Ejemplo 1. Ejmeplo

```
<?php
$array = array(
    'fruit1' => 'apple',
    'fruit2' => 'orange',
    'fruit3' => 'grape',
    'fruit4' => 'apple',
    'fruit5' => 'apple');

// this cycle echoes all associative array
// key where value equals "apple"
while ($fruit_name = current($array)) {
    if ($fruit_name == 'apple') {
        echo key($array).'<br />';
    }
    next($array);
}
?>
```

Vea también: **current()**, **next()**

krsort

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

krsort -- Ordena una matriz por clave en orden inverso

Descripción

int **krsort** (array &matriz [, int sort_flags])

Ordena una matriz por clave en orden inverso, manteniendo las correlaciones clave a dato. Esto es útil principalmente en matrices asociativas.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplo 1. Ejemplo

```
<?php
$fruits = array("d"=>"lemon", "a"=>"orange", "b"=>"banana", "c"=>"apple");
krsort($fruits);
reset($fruits);
while (list($key, $val) = each($fruits)) {
    echo "$key = $val\n";
}
?>
```

El resultado del ejemplo seria:

```
d = lemon
c = apple
b = banana
a = orange
```

Puede modificar el comportamiento del ordenamiento usando el parámetro opcional *sort_flags*, para más detalles vea **sort()**.

Vea también: **asort()**, **arsort()**, **ksort()**, **sort()**, **natsort()**, **rsort()**.

ksort

(PHP 3, PHP 4 , PHP 5)

ksort -- Ordena una matriz por clave

Descripción

int **ksort** (array &matriz [, int sort_flags])

Ordena una matriz por clave, manteniendo las correlaciones clave a dato. Esto es útil principalmente en matrices asociativas.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplo 1. Ejemplo

```
<?php
$fruits = array("d"=>"lemon", "a"=>"orange", "b"=>"banana", "c"=>"apple");
ksort($fruits);
reset($fruits);
while (list($key, $val) = each($fruits)) {
    echo "$key = $val\n";
}
?>
```

El resultado del ejemplo seria:

```
a = orange
b = banana
c = apple
d = lemon
```

Puede modificar el comportamiento del ordenamiento usando el parÃ‚Â´metro opcional *sort_flags*, para más detalles vea **sort()**.

Vea también: **asort()**, **arsort()**, **krsort()**, **uksort()**, **sort()**, **natsort()** y **rsort()**.

list

(PHP 3, PHP 4, PHP 5)

list -- Asigna variables como si fueran una matriz

Descripción

void **list** (mixed varname, mixed ...)

Como **array()**, esta no es realmente una función, sino una construcción del lenguaje. **list()** se usa para asignar una lista de variables en una sola operación.

> **Nota: list()** solo funciona con matrices numéricas y asume que el índice numérico empieza en 0.

Ejemplo 1. Ejemplo de list()

```
<?php

$info = array('coffee', 'brown', 'caffeine');

// Listing all the variables
list($drink, $color, $power) = $info;
echo "$drink is $color and $power makes it special.\n";

// Listing some of them
list($drink, , $power) = $info;
echo "$drink has $power.\n";

// Or let's skip to only the third one
list( , , $power) = $info;
echo "I need $power!\n";

?>
```

Ejemplo 2. Otro ejemplo del uso de list()

```
<table>
 <tr>
  <th>Employee name</th>
  <th>Salary</th>
 </tr>

<?php

$result = mysql_query("SELECT id, name, salary FROM employees", $conn);
while (list($id, $name, $salary) = mysql_fetch_row($result)) {
    echo " <tr>\n" .
          "  <td><a href=\"info.php?id=$id\">$name</a></td>\n" .
          "  <td>$salary</td>\n" .
          " </tr>\n";
}

?>

</table>
```

Aviso
list() asigna los valores comenzando por el parámetro más a la derecha. Si está usando variables independientes, no tiene que preocuparse acerca de esto. Pero si está usando matrices con índices usualmente se esperaría tener el orden de los índices en la matriz en la misma forma que se se escribió en la función **list()** de izquierda a derecha; lo cuál no ocurrirá, son asignados en orden inverso.

Ejemplo 3. Usando list() con matrices conínidces

```
<?php

$info = array('coffee', 'brown', 'caffeine');

list($a[0], $a[1], $a[2]) = $info;

var_dump($a);

?>
```

Da la siguiente salida (note el orden de los elementos comparado en el orden en el que fueron escritos en la función **list()**):

```
array(3) {
  [2]=>
  string(8) "caffeine"
  [1]=>
  string(5) "brown"
  [0]=>
  string(6) "coffee"
}
```

Vea también: **each()**, **array()** y **extract()**.

natcasesort

(PHP 4 , PHP 5)

natcasesort -- Ordena una matriz usando un algoritmo de "orden natural" sin distinguir mayúsculas de minúsculas

Descripción

void **natcasesort** (array &matriz)

Esta función implementa un algoritmo de ordenamiento que ordena cadenas alfanuméricas en la forma en que un ser humano lo haría, al mismo tiempo que conserva las asociaciones clave/valor. Esta propiedad se conoce como "ordenamiento natural".

natcasesort() es una versión de **natsort()** que no distingue entre mayúsculas y minúsculas.

Ejemplo 1. Ejemplo de natcasesort()

```
<?php
$matriz1 = $matriz2 = array('IMG0.png', 'img12.png', 'img10.png', 'img2.png', 'img1.png

sort($matriz1);
echo "Ordenamiento est&aacute;ndar\n";
print_r($matriz1);

natcasesort($matriz2);
echo "\nOrden natural (insensible a may&uacute;sculas y min&uacute;sculas)\n";
print_r($array2);
?>
```

El código anterior generará la siguiente salida:

```
Ordenamiento est&aacute;ndar
Array
(
    [0] => IMG0.png
    [1] => IMG3.png
    [2] => img1.png
    [3] => img10.png
    [4] => img12.png
    [5] => img2.png
)

Orden natural (insensible a may&uacute;sculas y min&uacute;sculas)
Array
(
    [0] => IMG0.png
    [4] => img1.png
    [3] => img2.png
    [5] => IMG3.png
    [2] => img10.png
    [1] => img12.png
)
```

Para más información vea: la página de Comparación de Cadenas en Orden Natural de Martin Pool. Vea también **sort()**, **natsort()**, **strnatcmp()**, y **strnatcasecmp()**.

natsort

(PHP 4 , PHP 5)

natsort -- Ordena una matriz usando un algoritmo de "orden natural"

Descripción

void **natsort** (array &matriz)

Esta función implementa un algoritmo que ordena cadenas alfanuméricas en la forma en que lo haría un ser humano, al mismo tiempo que conserva las asociaciones clave/valor. Esta propiedad es conocida como "ordenamiento natural". Un ejemplo de la diferencia entre éste y el algoritmo computacional normal de ordenamiento de cadenas (usado en **sort()**) puede apreciarse a continuación:

Ejemplo 1. Ejemplo de natsort()

```
<?php
$matriz1 = $matriz2 = array("img12.png", "img10.png", "img2.png", "img1.png");

sort($matriz1);
echo "Ordenamiento est&aacute;ndar\n";
print_r($matriz1);

natsort($matriz2);
echo "\nOrdenamiento natural\n";
print_r($matriz2);
?>
```

El anterior fragmento de código genera la siguiente salida:

```
Ordenamiento est&aacute;ndar
Array
(
    [0] => img1.png
    [1] => img10.png
    [2] => img12.png
    [3] => img2.png
)

Ordenamiento natural
Array
(
    [3] => img1.png
    [2] => img2.png
    [1] => img10.png
    [0] => img12.png
)
```

Para más información vea: la página de Comparación de Cadenas en Orden Natural de Martin Pool. Vea también **natcasesort()**, **strnatcmp()**, y **strnatcasecmp()**.

next

(PHP 3, PHP 4 , PHP 5)

next -- Avanza el puntero interno de una matriz

Descripción

mixed **next** (array &matriz)

Devuelve el elemento de la matriz que ocupa el lugar siguiente al apuntado por el puntero interno, o **FALSE** si no hay más elementos.

next() se comporta como **current()**, con una diferencia. Avanza el puntero interno de la matriz en una posición antes de devolver el elemento. Eso significa que devuelve el siguiente elemento de la matriz y que avanza el puntero interno en uno. Si al avanzar se pasa del final de la lista de elementos, **next()** devuelve **FALSE**.

Aviso
Si la matriz contiene elementos vacíos, esta función también devolverá **FALSE** para dichos elementos. Para recorrer adecuadamente una matriz que pueda contener elementos vacíos, vea la función **each()**.

Ejemplo 1. Ejemplo del uso de next()

```
<?php
$transport = array('foot', 'bike', 'car', 'plane');
$mode = current($transport); // $mode = 'foot';
$mode = next($transport);    // $mode = 'bike';
$mode = next($transport);    // $mode = 'car';
$mode = prev($transport);    // $mode = 'bike';
$mode = end($transport);     // $mode = 'plane';
?>
```

Vea también: **current()**, **end()** **prev()** y **reset()**

pos

pos -- Alias de **current()**

Descripción

Esta función es un alias de **current()**.

prev

(PHP 3, PHP 4 , PHP 5)

prev -- Rebobina el puntero interno de una matriz

Descripción

mixed **prev** (array &matriz)

Devuelve el elemento de la matriz que está en la posición anterior a la que apuntaba previamente el puntero interno, o **FALSE** si no hay más elementos.

Aviso
Si la matriz contiene elementos vacíos, esta función también devolverá **FALSE** para dichos elementos. Para recorrer adecuadamente una matriz que puede contener elementos vacíos, vea la función **each()**.

prev() se comporta igual que **next()**, excepto que rebobina el puntero interno una posición en lugar de avanzarlo.

Ejemplo 1. Ejemplo del uso de prev()

```
<?php
$transport = array('foot', 'bike', 'car', 'plane');
$mode = current($transport); // $mode = 'foot';
$mode = next($transport);    // $mode = 'bike';
$mode = next($transport);    // $mode = 'car';
$mode = prev($transport);    // $mode = 'bike';
$mode = end($transport);     // $mode = 'plane';
?>
```

Vea también: **current()**, **end()** **next()** y **reset()**.

range

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

range -- Crea una matriz que contiene un rango de elementos

Descripción

array **range** (number bajo, number alto [, number paso])

rango() devuelve una matriz de elementos desde *bajo* hasta *alto*, ambos inclusive. Si bajo > alto, la secuencia será del mayor al menor.

> **Nuevo parámetro:** El parámetro opcional *paso* fue añadido en 5.0.0.

Si un valor *paso* es dado, éste será usado como el incremento entre elementos en la secuencia. *paso* debería ser definido como un número positivo. Si no se especifica, *paso* tendrá un valor predeterminado de 1.

Ejemplo 1. Ejemplos de range()

```
<?php
// array(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12)
foreach (range(0, 12) as $numero) {
    echo $numero;
}

// El parametro paso fue introducido en 5.0.0
// array(0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100)
foreach (range(0, 100, 10) as $numero) {
    echo $numero;
}

// Uso de secuencias de caracteres introducidas en 4.1.0
// array('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i');
foreach (range('a', 'i') as $letra) {
    echo $letra;
}
// array('c', 'b', 'a');
foreach (range('c', 'a') as $letra) {
    echo $letra;
}
?>
```

> **Nota:** Antes de PHP 4.1.0, **range()** sólo generaba matrices de enteros incrementales. El soporte para secuencias de caracteres y matrices en decremento fue añadido en 4.1.0. Los valores de secuencia de caracteres esán limitados a una longitud de uno. Si una longitud superior a uno es ingresada, solo se usa el primer caracter.

Atención
En versiones de PHP desde 4.1.0 hasta 4.3.2, **range()** considera las cadenas numéricas como cadenas y no enteros. En su lugar, ellas serán usadas para secuencias de caracteres. Por ejemplo, *"4242"* es tratado como *"4"*.

Vea también **shuffle()**, **array_fill()**, y foreach.

reset

(PHP 3, PHP 4 , PHP 5)

reset -- Fija el puntero interno de una matriz a su primer elemento

Descripción

mixed **reset** (array &matriz)

reset() rebobina el puntero interno de la *matriz* a su primer elemento. Y regresa el valor de ese elemento, o **FALSE** si la matriz está vacía.

Ejemplo 1. Ejemplo de reset()

```
<?php

$array = array('step one', 'step two', 'step three', 'step four');

// by default, the pointer is on the first element
echo current($array) . "<br />\n"; // "step one"

// skip two steps
next($array);
next($array);
echo current($array) . "<br />\n"; // "step three"

// reset pointer, start again on step one
reset($array);
echo current($array) . "<br />\n"; // "step one"

?>
```

Vea también: **current()**, **each()**, **next()** y **prev()**.

rsort

(PHP 3, PHP 4 , PHP 5)

rsort -- Ordena una matriz en orden inverso

Descripción

void **rsort** (array &matriz [, int sort_flags])

Esta función ordena una matriz en orden inverso (mayor a menor).

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplo 1. Ejemplo de rsort()

```
<?php
$fruits = array("lemon", "orange", "banana", "apple");
rsort($fruits);
reset($fruits);
while (list($key, $val) = each($fruits)) {
    echo "$key = $val\n";
}
?>
```

El resultado del ejemplo seria:

```
0 = orange
1 = lemon
2 = banana
3 = apple
```

Las frutas han sido ordenadas en orden alfabético inverso.

Puede modificar el comportamiento del orden usando el parámetro opcional *sort_flags*, para más detalles vea **sort()**.

Vea también: **arsort()**, **asort()**, **ksort()**, **sort()**, y **usort()**.

shuffle

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

shuffle -- Mezcla una matriz

Descripción

void **shuffle** (array &matriz)

Esta función mezcla (cambia aleatoriamente el orden de los elementos de) una matriz.

Ejemplo 1. Ejemplo de shuffle()

```
<?php
$numeros = range (1,20);
srand (time());
shuffle ($numeros);
while (list(, $numero) = each ($numeros)) {
    echo "$numero ";
}
?>
```

> **Nota:** A partir de PHP 4.2.0, no es necesario inicializar el generador de números aleatorios con **srand()** ó **mt_srand()**, ya que esto se hace ahora automáticamente.

Vea también: **arsort()**, **asort()**, **ksort()**, **rsort()**, **sort()** y **usort()**.

sizeof

sizeof -- Alias de **count()**

Descripción

Esta función es un alias de **count()**.

sort

(PHP 3, PHP 4 , PHP 5)

sort -- Ordena una matriz

Descripción

void **sort** (array &matriz [, int sort_flags])

Esta función ordena una matriz. Los elementos estarán ordenados de menor a mayor cuando la función termine.

> **Nota:** Esta función asigna nuevos índices en *matriz*. Esto quitará cualquier índice existente que se haya aignado, en vez de solo reordenar los índices.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplo 1. Ejemplo de sort()

```
<?php

$fruits = array("lemon", "orange", "banana", "apple");
sort($fruits);
reset($fruits);
while (list($key, $val) = each($fruits)) {
    echo "fruits[" . $key . "] = " . $val . "\n";
}

?>
```

El resultado del ejemplo seria:

```
fruits[0] = apple
fruits[1] = banana
fruits[2] = lemon
fruits[3] = orange
```

Las frutas han sido ordenadas en orden alfabético.

El parámetro opcional *sort_flags* puede ser usado para modificar el comportamiento del ordenamiento usando estos valores:

Tipos de banderas para el ordenamiento:

- **SORT_REGULAR** - comparación normal (no cambia los tipos)
- **SORT_NUMERIC** - comparación numérica
- **SORT_STRING** - comparación por cadenas
- **SORT_LOCALE_STRING** - compara elementos como cadenas, basado en la posición actual. Agregado en PHP 5.0.2.

> **Nota:** El segundo parámetro fue agregado en PHP 4.

Aviso
Tenga cuidado cuando ordene matrices con tipos de datos mixtos porque **sort()** puede producir resultados impredecibles.

Vea también: **arsort()**, **asort()**, **ksort()**, **natsort()**, **natcasesort()**, **rsort()**, **usort()**, **array_multisort()** y **uksort()**.

uasort

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

uasort -- Ordena una matriz mediante una función de comparación definida por el usuario y mantiene la asociación de índices

Descripción

void **uasort** (array &matriz, function func_comparar)

Esta función ordena una matriz de modo que los índices de la misma mantengan su correlación con los elementos a los que están asociados. Esto se utiliza principalmente para ordenar matrices asociativas en las que el orden de los elementos es importante. La función de comparación es definida por el usuario.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

> **Nota:** Vea por favor **usort()** y **uksort()** para ejemplos de funciones de comparación definidas por el usuario.

Vea tambié **usort()**, **uksort()**, **sort()**, **asort()**, **arsort()**, **ksort()** y **rsort()**.

uksort

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

uksort -- Ordena una matriz por claves mediante una función definida por el usuario

Descripción

void **uksort** (array &matriz, function func_comparar)

Esta función ordenará las claves de una matriz utilizando una función de comparación suministrada por el usuario. Si la matriz a ordenar necesita utilizar un criterio poco trivial, esta es la función que deberá usar.

Function *cmp_function* should accept two parameters which will be filled by pairs of *array* keys. The comparison function must return an integer less than, equal to, or greater than zero if the first argument is considered to be respectively less than, equal to, or greater than the second.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplo 1. Ejemplo de uksort()

```
<?php
function cmp($a, $b)
{
    if ($a == $b) {
        return 0;
    }
    return ($a > $b) ? -1 : 1;
}

$a = array(4 => "four", 3 => "three", 20 => "twenty", 10 => "ten");

uksort($a, "cmp");

while (list($key, $value) = each($a)) {
    echo "$key: $value\n";
}
?>
```

El resultado del ejemplo seria:

```
20: twenty
10: ten
4: four
3: three
```

Vea también: **usort()**, **uasort()**, **sort()**, **asort()**, **arsort()**, **ksort()**, **natsort()**, y **rsort()**.

usort

(PHP 3>= 3.0.3, PHP 4 , PHP 5)

usort -- Ordena una matriz por sus valores usando una función de comparación definida por el usuario

Descripción

bool **usort** (array &matriz, callback funcion_comp)

Esta función ordenará una matriz por sus valores usando una función de comparación definida por el usuario. Si la matriz que desea ordenar necesita ser ordenada mediante ciertos criterios especiales, es buena idea usar esta función.

La función de comparación debe devolver un entero menor que, igual, o mayor que cero si el primer argumento es considerado menor, igual, o mayor que el segundo, respectivamente.

> **Nota:** Si dos miembros son comparados como iguales, su orden en la matriz resultante es indefinido. Hasta PHP 4.0.6, las funciones definidas por el usuario mantenían el orden original para esos elementos, pero con el nuevo algoritmo de ordenamiento introducido en 4.1.0 este ya no es el caso, ya que no hay forma de hacerlo de manera eficiente.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplo 1. Ejemplo de usort()

```
<?php
function cmp($a, $b)
{
    if ($a == $b) {
        return 0;
    }
    return ($a < $b) ? -1 : 1;
}

$a = array(3, 2, 5, 6, 1);

usort($a, "cmp");

while (list($clave, $valor) = each($a)) {
    echo "$clave: $valor\n";
}
?>
```

Este ejemplo produciría la salida:

```
0: 1
1: 2
2: 3
3: 5
4: 6
```

Nota: Obviamente en este ejemplo trivial, la función **sort()** sería más apropiada.

Ejemplo 2. Ejemplo de usort() usando una matriz multi-dimensional

```
<?php
function cmp($a, $b)
{
    return strcmp($a["fruta"], $b["fruta"]);
}

$frutas[0]["fruta"] = "limones";
$frutas[1]["fruta"] = "bananos";
$frutas[2]["fruta"] = "granadillas";

usort($frutas, "cmp");

while (list($clave, $valor) = each($frutas)) {
    echo "\$frutas[$clave]: " . $valor["fruta"] . "\n";
}
?>
```

Cuando se ordena una matriz multi-dimensional, *$a* y *$b* contienen referencias al primer índice de la matriz.

Este ejemplo produciría la salida:

```
$frutas[0]: bananos
$frutas[1]: granadillas
$frutas[2]: limones
```

Ejemplo 3. Ejemplo de usort() usando una función miembro de un objeto

```
<?php
class ObjPrueba {
    var $nombre;

    function ObjPrueba($nombre)
    {
        $this->nombre = $nombre;
    }

    /* Esta es la funcion estatica de comparacion: */
    function cmp_obj($a, $b)
    {
        $al = strtolower($a->nombre);
        $bl = strtolower($b->nombre);
        if ($al == $bl) {
            return 0;
        }
        return ($al > $bl) ? +1 : -1;
    }
}

$a[] = new ObjPrueba("c");
$a[] = new ObjPrueba("b");
$a[] = new ObjPrueba("d");

usort($a, array("ObjPrueba", "cmp_obj"));

foreach ($a as $item) {
    echo $item->nombre . "\n";
}
?>
```

Este ejemplo produciría la salida:

```
b
c
d
```

Vea también **uasort()**, **uksort()**, **sort()**, **asort()**, **arsort()**,**ksort()**, **natsort()**, and **rsort()**.

IV. Funciones Aspell [deprecated]

Introducción

La función **aspell()** permite comprobar la ortografía de una palabra y ofrece alternativas a la misma.

> **Nota:** Esta extensión ha sido removida de *PHP* y no se encuentra disponible a partir de PHP 4.3.0. Si quereis utilizar capacidades de comprobación ortográfica en php, usar pspell. Utiliza la biblioteca pspell y funciona con versiones recientes de aspell.

Requirimientos

aspell funciona solamente con versiones muy antiguas (hasta la .27.* mas ó menos) de la biblioteca aspell. Ni este módulo ni las versiones de la biblioteca aspell se soportan actualmente. Necesitais la biblioteca aspell, disponible en : http://aspell.sourceforge.net/.

Instalación

En PHP 4, estas funciones se encuentran disponibles únicamente si PHP fue configurado con *--with-aspell=[DIR]*.

Ver también

Ver tambien pspell.

Tabla de contenidos

aspell_check-raw

(PHP 3>= 3.0.7, PHP 4 <= 4.2.3)

aspell_check-raw -- Comprueba una palabra sin cambiarla ó intentar arreglarla [deprecated]

Descripción

boolean **aspell_check_raw** (int dictionary_link, string word)

aspell_check_raw() comprueba la ortografía de una palabra,sin cambiarla ni intentar arreglarla esté bien o mal. Si está bien, devuelve cierto (**TRUE**), si no lo está, devuelve falso(**FALSE**).

Ejemplo 1. aspell_check_raw
```
$aspell_link = aspell_new("english");

if (aspell_check_raw($aspell_link, "test")) {
    echo "This is a valid spelling";
} else {
    echo "Sorry, wrong spelling";
}
```

aspell_check

(PHP 3>= 3.0.7, PHP 4 <= 4.2.3)

aspell_check -- Comprueba una palabra[deprecated]

Descripción

boolean **aspell_check** (int dictionary_link, string word)

aspell_check() comprueba la ortografía de una palabra, y devuelve cierto(**TRUE**) si la ortografía es correcta ,falso (**FALSE**) si no lo es .

Ejemplo 1. aspell_check

```
$aspell_link = aspell_new("english");

if (aspell_check($aspell_link, "testt")) {
    echo "This is a valid spelling";
} else {
    echo "Sorry, wrong spelling";
}
```

aspell_new

(PHP 3>= 3.0.7, PHP 4 <= 4.2.3)

aspell_new -- Lee un nuevo diccionario [deprecated]

Descripción

int **aspell_new** (string master, string personal)

aspell_new() Abre un nuevo diccionario devolviendo el identificador de este para ser utilizado en otras funciones ortográficas.

Ejemplo 1. Nuevo_diccionario

```
$aspell_link = aspell_new("english");
```

aspell_suggest

(PHP 3>= 3.0.7, PHP 4 <= 4.2.3)

aspell_suggest -- Sugiere la ortografía para una palabra [deprecated]

Descripción

array **aspell_suggest** (int dictionary_link, string word)

aspell_suggest() devuelve una matriz con posibles correciones ortográficas para la palabra dada.

Ejemplo 1. aspell_suggest

```
$aspell_link = aspell_new("english");

if (!aspell_check($aspell_link, "test")) {
    $suggestions = aspell_suggest($aspell_link, "test");

    foreach ($suggestions as $suggestion) {
        echo "Possible spelling: $suggestion<br>\n";
    }
}
```

V. Funciones matemáticas de precisión arbitraria BCMath

Introducción

Para operaciones matemáticas de precisión arbitraria, PHP tiene disponible la Calculadora Binaria que soporta números de cualquier tamaño y precisión, representados como cadenas de texto.

Requirimientos

Desde PHP 4.0.4, libbcmath se encuentra incorporada en PHP. No se necesitan bibliotecas externas para esta extensión.

Instalación

Estas funciones solo están disponibles si PHP fue configurado con *--enable-bcmath*. En PHP 3, estas funciones sólo están disponibles si PHP *no* fue configurado con *--disable-bcmath*.

La versión para Windows de *PHP* tiene soporte nativo para esta extensión. No se necesita cargar ninguna extensión adicional para usar estas funciones.

Configuración en tiempo de ejecución

El comportamiento de estas funciones está afectado por los valores definidos en `php.ini`.

Tabla 1. Opciones de configuración de BCMath

Nombre	Por defecto	Modificable
bcmath.scale	0	PHP_INI_ALL

Para más detalles y la definición de las constantes PHP_INI_*, consulte **ini_set()**.

A continuación se presenta una corta explicación de las directivas de configuración.

bcmath.scale **integer**

Número de dígitos decimales para todas las funciones de bcmath. Vea también **bcscale()**.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Esta extensión no tiene ninguna constante definida.

Tabla de contenidos

bcadd

(PHP 3, PHP 4 , PHP 5)

bcadd -- Suma dos números de precisión arbitriaria.

Descripción

string **bcadd** (string operando izq, string operando der [, int escala])

Suma el *operando izq* con el *operando der* y devuelve la suma en una cadena de texto. El parámetro opcional *escala* se usa para fijar el número de dígitos tras el punto decimal que aparecerán en el resultado.

Ejemplo 1. Ejemplo bcadd()

```
<?php

$a = 1.234;
$b = 5;

echo bcadd($a, $b);      // 6
echo bcadd($a, $b, 4);   // 6.2340

?>
```

Vea también **bcsub()**.

bccomp

(PHP 3, PHP 4 , PHP 5)

bccomp -- Compara dos números de precisión arbitraria.

Descripción

int **bccomp** (string operando izq, string operando der [, int escala])

Compara el *operando izq* con el *operando der* y devuelve el resultado como un entero. El parámetro opcional *escala* se usa para fijar el número de dígitos tras el punto decimal que se utilizarán en la comparación. El valor devuelto es 0 si los dos operandos son iguales. Si el *operando izq* es mayor que el *operando der* el valor devuelto es +1 y si el *operando izq* es menor que el *operando der* el valor devuelto es -1.

Ejemplo 1. Ejemplo bccomp()

```
<?php
echo bccomp('1', '2') . "\n";

echo bccomp('1.00001', '1', 3) . "\n";
echo bccomp('1.00001', '1', 5);
?>
```

Resultado de este ejemplo:

```
-1
0
1
```

bcdiv

(PHP 3, PHP 4 , PHP 5)

bcdiv -- Divide dos números de precisión arbitraria.

Descripción

string **bcdiv** (string operando izq, string operando der [, int escala])

Divide el *operando izq* por el *operando der* y devuelve el resultado. El parámetro opcional *escala* fija el número de dígitos tras el punto decimal a usar en el resultado.

Ejemplo 1. Ejemplo bcdiv()

```
<?php

echo bcdiv(105, 6.55957, 3);  // 16.007

?>
```

Ver también **bcmul()**.

bcmod

(PHP 3, PHP 4 , PHP 5)

bcmod -- Obtiene el módulo de un número de precisión arbitraria.

Descripción

string **bcmod** (string operando izq, string modulo)

Obtiene el módulo del *operando izq* usando *modulo*.

Ejemplo 1. Ejemplo bcmod()

```
<?php
echo bcmod(4, 2) . "\n";
echo bcmod(2, 4);
?>
```
Resultado de este ejemplo:
```
0
2
```
Ver también **bcdiv()**.

bcmul

(PHP 3, PHP 4 , PHP 5)

bcmul -- Multiplica dos números de precisión arbitraria.

Descripción

string **bcmul** (string operando izq, string operando der [, int escala])

Multiplica el *operando izq* por el *operando der* y devuelve el resultado. El parámetro opcional *escala* fija el número de dígitos tras el punto decimal del resultado.

Ejemplo 1. Ejemplo bcmul()
```
<?php
echo bcmul(1.34747474747, 35, 3) . "\n";
echo bcmul(2, 4);
?>
```
Resultado de este ejemplo:
```
47.162
8
```
Ver también **bcdiv()**.

bcpow

(PHP 3, PHP 4 , PHP 5)

bcpow -- Eleva un número de precisión arbitraria a otro.

Descripción

string **bcpow** (string x, string y [, int escala])

Eleva *x* a la potencia de *y*. El parámetro opcional *escala* se puede usar para fijar el número de dígitos tras el punto decimal del resultado.

Ejemplo 1. Ejemplo bcpow()
```
<?php

echo bcpow(4.2, 3, 2); // 74.08

?>
```
Ver también **bcpowmod()** y **bcsqrt()**

bcpowmod

(PHP 5)

bcpowmod -- Eleva un número de precisión arbitraria a otro, reducido por un módulo especificado

Descripción

string **bcpowmod** (string x, string y, string módulo [, int escala])

Utiliza el método de exponenciación-rápida para elevar *x* a la potencia *y* con respecto al módulo *módulo*. La *escala* opcional puede ser usada para establecer el número de dígitos después de la coma decimal en el resultado.

> **Nota:** Dado que éste método usa la operación de módulo, los números no-naturales pueden dar resultados inesperados. Un número natural es cualquier entero positivo diferente de cero.

Ejemplos

Las siguientes dos declaraciones son idénticas funcionalmente. Sin embargo, la versión que usa **bcpowmod()** se ejecuta en menos tiempo y puede aceptar parámetros más grandes.

```
<?php
$a = bcpowmod($x, $y, $mod);

$b = bcmod(bcpow($x, $y),$mod);

// $a y $b poseen valores iguales.
?>
```

Ver también

bcpow(), y **bcmod()**.

bcscale

(PHP 3, PHP 4 , PHP 5)

bcscale -- Fija el parámetro de escala por defecto para todas las funciones matemáticas bc.

Descripción

bool **bcscale** (int escala)

Esta función fija el parámetro de escala por defecto para las subsiguientes funciones matemáticas bc que no especifican dicho parámetro explícitamente. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplo 1. Ejemplo bcscale()

```php
<?php

// default scale : 3
bcscale(3);
echo bcdiv(105, 6.55957); // 16.007

// this is the same without bcscale()
echo bcdiv(105, 6.55957, 3); // 16.007

?>
```

bcsqrt

(PHP 3, PHP 4 , PHP 5)

bcsqrt -- Obtiene la raíz cuadrada de un número de precisión arbitraria.

Descripción

string **bcsqrt** (string operando, int escala)

Devuelve la raíz cuadrada del *operando*. El parámetro opcional *escala* fija el número de dígitos tras el punto decimal del resultado.

Ejemplo 1. Ejemplo bcsqrt()

```php
<?php

echo bcsqrt(2, 3); // 1.414

?>
```

Ver también **bcpow()**.

bcsub

(PHP 3, PHP 4 , PHP 5)

bcsub -- Resta un número de precisión arbitraria de otro.

Descripción

string **bcsub** (string operando izq, string operando der [, int escala])

Resta el *operando der* del *operando izq* y devuelve el resultado en una cadena. El parámetro opcional *escala* se utiliza para fijar el número de dígitos tras el punto decimal del resultado.

Ejemplo 1. Ejemplo bcsub()

```php
<?php

$a = 1.234;
$b = 5;

echo bcsub($a, $b);     // -3
echo bcsub($a, $b, 4);  // -3.7660

?>
```

Ver también **bcadd()**.

VI. PHP bytecode Compiler

Introducción

Aviso
Esta extensión es *EXPERIMENTAL*. Esto significa que el comportamiento de esta extensión, los nombre de sus funciones y en definitiva TODO lo documentado sobre esta extensión, puede cambiar en una futura versión de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Bcompiler was written for two reasons:

To encode entire script in a proprietary PHP application

To encode some classes and/or functions in a proprietary PHP application

To enable the production of php-gtk applications that could be used on client desktops, without the need for a php.exe.

To do the feasibility study for a PHP to C converter

The first of these goals is achieved using the **bcompiler_write_header()**, **bcompiler_write_file()** and **bcompiler_write_footer()** functions. The bytecode files can be written as either uncompressed or plain. To use the generated bytecode, you can simply include it with include or require statements.

The second of these goals is achieved using the **bcompiler_write_header()**, **bcompiler_write_class()**, **bcompiler_write_footer()**, **bcompiler_read()**, and **bcompiler_load()** functions. The bytecode files can be written as either uncompressed or plain. The **bcompiler_load()** reads a bzip compressed bytecode file, which tends to be 1/3 of the size of the original file.

To create EXE type files, bcompiler has to be used with a modified sapi file or a version of PHP which has been compiled as a shared library. In this scenario, bcompiler reads the compressed bytecode from the end of the exe file.

bcompiler can improve performance by about 30% when used with uncompressed bytecodes only. But keep in mind that uncompressed bytecode can be up to 5 times larger than the original source code. Using bytecode compression can save your space, but decompression requires much more time than parsing a source. bcompiler also does not do any bytecode optimization, this could be added in the future...

In terms of code protection, it is safe to say that it would be impossible to recreate the exact source code that it was built from, and without the accompanying source code comments. It would effectively be useless to use the bcompiler bytecodes to recreate and modify a class. However it is possible to retrieve data from a bcompiled bytecode file - so don't put your private passwords or anything in it.

Instalación

short installation note:

- You need at least PHP 4.3. for the compression to work

- To install on PHP 4.3 and later at the unix command prompt type **pear install bcompiler**
- To install on Windows, until the binary package distribution mechanism is finished please search the archives of the pear-general mailing list for pre-built packages. (or send an email to it if you could not find a reference)
- To install on older versions you need to make some slight changes to the build.
- untar the `bcompiler.tgz` archive into `php4/ext`.(Get it directly from PECL http://pecl.php.net/get/bcompiler)
- If the new directory is now called something like bcompiler-0.x, then you should rename it to bcompiler (except you only want to build it as self-contained php-module).
- If you are using versions before PHP 4.3, the you will need to copy the `Makefile.in.old` to `Makefile.in` and `config.m4.old` to `config.m4`.
- run **phpize** in `ext/bcompiler`
- run **./buildconf** in `php4`
- run **configure** with *--enable-bcompiler* (and your other options)
- **make; make install**
- that's it.

Contact Information

If you have comments, bugfixes, enhancements or want to help developing this beast, you can drop me a mail at alan_k@php.net. Any help is very welcome.

Tabla de contenidos

bcompiler_load_exe

(no version information, might be only in CVS)

bcompiler_load_exe -- Reads and creates classes from a bcompiler exe file

Description

bool **bcompiler_load_exe** (string filename)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Reads data from a bcompiler exe file and creates classes from the bytecodes

Ejemplo 1. bcompiler_load() example

```
<?php

bcompiler_load_exe("/tmp/example.exe");
print_r(get_defined_classes());

?>
```

bcompiler_load

(no version information, might be only in CVS)

bcompiler_load -- Reads and creates classes from a bz compressed file

Description

bool **bcompiler_load** (string filename)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

> **Nota:** Please use include or require statements to parse bytecodes, it's more portable and convenient way than using this function.

Reads data from a bzcompressed file and creates classes from the bytecodes. Please note that this function won't execute script body code contained in the bytecode file.

Ejemplo 1. bcompiler_load() example

```
<?php

bcompiler_load("/tmp/example");

print_r(get_defined_classes());

?>
```

bcompiler_parse_class

(no version information, might be only in CVS)

bcompiler_parse_class -- Reads the bytecodes of a class and calls back to a user function

Description

bool **bcompiler_parse_class** (string class, string callback)

> **Nota:** This function has been removed from bcompiler and is no longer available as of bcompiler 0.5.

reads the bytecodes of a class and calls back to a user function

Ejemplo 1. bcompiler_parse_class() example

```
<?php

function readByteCodes($data) {
  print_r($data);
}

bcompiler_parse_class("DB","readByteCodes");

?>
```

bcompiler_read

(no version information, might be only in CVS)

bcompiler_read -- Reads and creates classes from a filehandle

Description

bool **bcompiler_read** (resource filehandle)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

> **Nota:** Please use include or require statements to parse bytecodes, it's more portable and convenient way than using this function.

Reads data from a open file handle and creates classes from the bytecodes. Please note that this function won't execute script body code contained in the bytecode file.

Ejemplo 1. bcompiler_read() example

```
<?php
$fh = fopen("/tmp/example","r");
bcompiler_read($fh);
fclose($fh);
print_r(get_defined_classes());

?>
```

bcompiler_write_class

(no version information, might be only in CVS)

bcompiler_write_class -- Writes an defined class as bytecodes

Description

bool **bcompiler_write_class** (resource filehandle, string className [, string extends])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

This reads the bytecodes from PHP for an existing class, and writes them to the open file handle, It does not perform dependency checking, so make sure you write the classes in an order that will not result in an 'undefined class' occurring when you load it.

Ejemplo 1. bcompiler_write_class() example

```
<?php
$fh = fopen("/tmp/example","w");
bcompiler_write_header($fh);
bcompiler_write_class($fh,"DB");
// you must write DB_common before DB_mysql, as DB_mysql extends DB_common.
bcompiler_write_class($fh,"DB_common");
bcompiler_write_class($fh,"DB_mysql");
bcompiler_write_footer($fh);
fclose($fh);

?>
```

See also **bcompiler_write_header()**, and **bcompiler_write_footer()**.

bcompiler_write_constant

(no version information, might be only in CVS)

bcompiler_write_constant -- Writes a defined constant as bytecodes

Description

bool **bcompiler_write_constant** (resource filehandle, string constantName)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

This function reads the bytecodes from PHP for an existing constant, and writes them to the open file handle.

Ejemplo 1. bcompiler_write_constant() example

```
<?php
define("MODULE_MAX", 30);

$fh = fopen("/tmp/example","w");
bcompiler_write_header($fh);
bcompiler_write_constant($fh,"MODULE_MAX");
bcompiler_write_footer($fh);
fclose($fh);

?>
```

See also **bcompiler_write_header()**, and **bcompiler_write_footer()**.

bcompiler_write_exe_footer

(no version information, might be only in CVS)

bcompiler_write_exe_footer -- Writes the the start pos, and sig to the end of a exe type file

Description

bool **bcompiler_write_exe_footer** (resource filehandle, int startpos)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

An EXE (or self executable) file consists of 3 parts,

The stub (executable code, e.g. a compiled C program) that loads PHP interpreter, bcompiler extension, stored Bytecodes and initiates a call for the specified function (e.g. main) or class method (e.g. main::main)

The Bytecodes (uncompressed only for the moment)

The bcompiler EXE footer

The *startpos* is the file position at which the Bytecodes start, and can be obtained using ftell($fh).

To obtain a suitable stub you can compile php_embed-based stub phpe.c located in the examples/embed directory on bcompiler's CVS.

Ejemplo 1. bcompiler_write_footer() example

```
<?php
/* creating the output file (example.exe) */
$fh = fopen("example.exe", "w");
/* 1) writing a stub (phpe.exe) */
$size = filesize("phpe.exe");
$fr = fopen("phpe.exe", "r");
fwrite($fh, fread($fr, $size), $size);
$startpos = ftell($fh);
/* 2) writing bytecodes */
bcompiler_write_header($fh);
bcompiler_write_class($fh, "myclass");
bcompiler_write_function($fh, "main");
bcompiler_write_footer($fh);
/* 3) writing EXE footer */
bcompiler_write_exe_footer($fh, $startpos);
/* closing the output file */
fclose($fh);
?>
```

See also **bcompiler_write_header()**, **bcompiler_write_class()**, and **bcompiler_write_footer()**.

bcompile_write_file

(no version information, might be only in CVS)

bcompile_write_file -- Writes a php source file as bytecodes

Description

bool **bcompiler_write_file** (resource filehandle, string filename)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

This function complies specified source file into bytecodes, and writes them to the open file handle.

Ejemplo 1. bcompiler_write_file() example

```
<?php
$fh = fopen("example.phb", "w");
bcompiler_write_header($fh);
bcompiler_write_file($fh, "example.php");
bcompiler_write_footer($fh);
fclose($fh);
/* the following should be ñalent:
include "example.php";
   and
include "example.phb";
*/
?>
```

See also **bcompiler_write_header()**, and **bcompiler_write_footer()**.

bcompiler_write_footer

(no version information, might be only in CVS)

bcompiler_write_footer -- Writes the single character \x00 to indicate End of compiled data

Description

bool **bcompiler_write_footer** (resource filehandle)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Writes the a single character \x00 to indicate End of compiled data

Ejemplo 1. bcompiler_write_footer() example

```
<?php
$fh = fopen("/tmp/example","w");
bcompiler_write_header($fh);
bcompiler_write_class($fh,"DB");
bcompiler_write_class($fh,"DB_common");
bcompiler_write_footer($fh);
fclose($fh);

?>
```

See also **bcompiler_write_header()**, and **bcompiler_write_header()**.

bcompiler_write_function

(no version information, might be only in CVS)

bcompiler_write_function -- Writes an defined function as bytecodes

Description

bool **bcompiler_write_function** (resource filehandle, string functionName)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

This reads the bytecodes from PHP for an existing function, and writes them to the open file handle. Order is not important, (eg. if function b uses function a, and you compile it like the example below, it will work perfectly OK)

Ejemplo 1. bcompiler_write_function() example

```
<?php
$fh = fopen("/tmp/example","w");
bcompiler_write_header($fh);
bcompiler_write_function($fh,"my_function_a");
bcompiler_write_function($fh,"my_function_b");
bcompiler_write_footer($fh);
fclose($fh);

?>
```

See also **bcompiler_write_header()**, and **bcompiler_write_footer()**.

bcompiler_write_functions_from_file

(no version information, might be only in CVS)

bcompiler_write_functions_from_file -- Writes all functions defined in a file as bytecodes

Description

bool **bcompiler_write_functions_from_file** (resource filehandle, string fileName)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

This function searches for all functions declared in the given file, and writes their correspondent bytecodes to the open file handle. Always remember to include/require the file you intend to compile.

Ejemplo 1. bcompiler_write_functions_from_file() example

```
<?php
require('module.php');

$fh = fopen("/tmp/example","w");
bcompiler_write_header($fh);
bcompiler_write_functions_from_file($fh,'module.php');
bcompiler_write_footer($fh);
fclose($fh);

?>
```

See also **bcompiler_write_header()**, and **bcompiler_write_footer()**.

bcompiler_write_header

(no version information, might be only in CVS)

bcompiler_write_header -- Writes the bcompiler header

Description

bool **bcompiler_write_header** (resource filehandle [, string write_ver])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Writes the header part of a bcompiler file. Optional second parameter can be used to write bytecode in a previously used format, so that you can use it with older versions of bcompiler.

Ejemplo 1. bcompiler_write_header() example

```
<?php
$fh = fopen("/tmp/example","w");
bcompiler_write_header($fh);
bcompiler_write_class($fh,"DB");
bcompiler_write_footer($fh);
fclose($fh);

?>
```

See also **bcompiler_write_file()**, **bcompiler_write_class()**, **bcompiler_write_function()**, and **bcompiler_write_footer()**.

VII. Funciones de compresión Bzip2

Introducción

Las funciones bzip2 son usadas para leer y escribir de forma transparente, ficheros comprimidos bzip2 (.bz2)

Requirimientos

Este módulo usa las funciones de la biblioteca bzip2 de Julian Seward. este módulo requiere bzip2/libbzip2 version >= 1.0.x.

Instalación

El soporte para bzip2 en PHP no está habilitado por defecto. Necesita usar el parámetro de configuración *--with-bz2[=DIR]* a la hora de compilar PHP para habilitar el soporte para bzip2.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

Esta extensión define un tipo de recurso: un puntero de fichero que identifica el fichero bz2 con el que se va a trabajar.

Constantes predefinidas

Esta extensión no tiene ninguna constante definida.

Ejemplos

Este ejemplo abre un fichero temporal, escribe una cadena literal en el y presenta el contenido de dicho fichero.

Ejemplo 1. Ejemplo simple de bzip2

```
<?php

$filename = "/tmp/testfile.bz2";
$str = "This is a test string.\n";

// open file for writing
$bz = bzopen($filename, "w");

// write string to file
bzwrite($bz, $str);

// close file
bzclose($bz);

// open file for reading
$bz = bzopen($filename, "r");

// read 10 characters
print bzread($bz, 10);

// output until end of the file (or the next 1024 char) and close it.
print bzread($bz);

bzclose($bz);

?>
```

Tabla de contenidos

bzclose

(PHP 4 >= 4.0.4, PHP 5)

bzclose -- Cierra un archivo bzip2

Descripción

int **bzclose** (resource bz)

Cierra el archivo bzip2 referenciado por el apuntador *bz*.

Lista de parámetros

bz

El apuntador del archivo. Este debe ser válido y debe apuntar a un archivo abierto exitosamente por **bzopen()**.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ver también

bzopen()

bzcompress

(PHP 4 >= 4.0.4, PHP 5)

bzcompress -- Comprime una cadena a una forma de datos codificados bzip2

Descripción

cadena **bzcompress** (cadena fuente [, int tamañ_de_bloque [, int factor]])

bzcompress() comprime la cadena dada y la regresa como datos codificados bzip2.

Lista de parámetros

fuente

La cadena a ser comprimida.

tamañ_de_bloque

Especifica el tamaño de bloque a ser usado durante la compresión y debe ser un número entre 1 y 9, dando 9 la mejor compresión, pero usando más recursos para hacerlo. *tamañ_de_bloque* tiebe como valor por defecto 4.

factor

Controla como la fase de compresión se comporta cuando se presenta el peor caso, altamente repetitivo de entrada de datos. El valor puede estar entre 0 y 250, siendo 0 un caso especial y 30 el valor por defecto.

Sin importar el *factor*, la salida generada es la misma.

Valores retornados

La cadena comprimida.

Ejemplos

Ejemplo 1. Comprimiendo datos

```
<?php
$str = "sample data";
$bzstr = bzcompress($str, 9);
echo $bzstr;
?>
```

Ver también

bzdecompress()

bzdecompress

(PHP 4 >= 4.0.4, PHP 5)

bzdecompress -- Descomprime datos codificados con bzip2

Descripción

string **bzdecompress** (cadena fuente [, int small])

bzdecompress() descomprime los datos dados que contengan datos codificados bzip2.

Lista de parámetros

fuente

La cadena a descomprimir.

small

Si **TRUE**, será usado un algoritmo de descompresión alternativo, el cuál usa menos memoria (la memoria máxima requerida baja a un valor alrededor de 2300K) pero trabaja a la mitad de velocidad.

Vea la documentación de bzip2 para más información acerca de esa característica.

Valores retornados

La cadena descomprimida.

Ejemplos

Ejemplo 1. Descomprimiendo una cadena

```
<?php
$start_str = "This is not an honest face?";
$bzstr = bzcompress($start_str);

echo "Compressed String: ";
echo $bzstr;
echo "\n<br />\n";

$str = bzdecompress($bzstr);
echo "Decompressed String: ";
echo $str;
echo "\n<br />\n";
?>
```

Ver también

bzcompress()

bzerrno

(PHP 4 >= 4.0.4, PHP 5)

bzerrno -- Regresa un número de error bzip2

Descripción

int **bzerrno** (resource bz)

Regresa el número de error de cualquier error bzip2 regresado por el apuntador del archivo dado.

Lista de parámetros

bz

El apuntador del archivo. Este debe ser válido y debe apuntar a un archivo abierto exitosamente por **bzopen()**.

Valores retornados

Regresa el número de error como un entero.

Ver también

bzerror()
bzerrstr()

bzerror

(PHP 4 >= 4.0.4, PHP 5)

bzerror -- Regresa el número y la cadena de error bzip2 en una matriz

Descripción

array **bzerror** (resource bz)

Regresa el número y cadena de error de cualquier error bzip2 regresado por el apuntador de archivo dado.

Lista de parámetros

bz

El apuntador de archivo. Este debe ser válido y debe apuntar a un archivo exitosamente abierto con **bzopen()**.

Valores retornados

Regresa una matriz asociativa, con el código de error en *errno*, y el mensaje de error en *errstr*.

Ejemplos

Ejemplo 1. Ejemplo de bzerror()

```
<?php
$error = bzerror($bz);

echo $error["errno"];
echo $error["errstr"];
?>
```

Ver también

bzerrno()
bzerrstr()

bzerrstr

(PHP 4 >= 4.0.4, PHP 5)

bzerrstr -- Regresa una cadena de error de bzip2

Descripción

string **bzerrstr** (resource bz)

Obtiene la cadena de error de cualquier error bzip2 regresado por el apuntador de archivo dado.

Lista de parámetros

bz

El apuntador del archivo. Este debe ser válido y debe apuntar a un archivo abierto exitosamente por **bzopen()**.

Valores retornados

Regresa una cadena que contiene el mensaje de error.

Ver también

bzerrno()
bzerror()

bzflush

(PHP 4 >= 4.0.4, PHP 5)

bzflush -- Fuerza la escritura de todo los datos en el buffer

Descripción

int **bzflush** (resource bz)

Fuerza la escritura de todos los datos bzip2 en el buffer para el archivo relacionado con el apuntador. *bz*.

Lista de parámetros

bz

El apuntador del archivo. Este debe ser válido y debe apuntar a un archivo abierto exitosamente por **bzopen()**.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ver también

bzread()
bzwrite()

bzopen

(PHP 4 >= 4.0.4, PHP 5)

bzopen -- Abre un archivo comprimido bzip2

Descripción

resource **bzopen** (string archivo, string modo)

bzopen() abre un archivo bzip2 (.bz2) para lectura o escritura.

Lista de parámetros

archivo

El nombre del archivo a utilizar.

modo

Similar a la función **fopen()** ('r' para lectura, 'w' para escritura, etc.).

Valores retornados

Si la apertura del archivo falla, **bzopen()** regresa **`FALSE`**, de otro modo regresa un apuntador al archivo abierto.

Ejemplos

Ejemplo 1. Ejemplo de bzopen()

```
<?php

$file = "/tmp/foo.bz2";
$bz = bzopen($file, "r") or die("Couldn't open $file for reading");

bzclose($bz);

?>
```

Ver también

bzclose()

bzread

(PHP 4 >= 4.0.4, PHP 5)

bzread -- Lectura segura de archivo bzip2 binario

Descripción

string **bzread** (resource bz [, int longitud])

bzread() lee del apuntador de archivo bzip2 dado.

La lectura se detiene cuando *longitud* (de datos sin comprimir) es alcanzada o se encuentra el fin de archivo (EOF), lo que ocurra primero.

Lista de parámetros

bz

El apuntador del archivo. Este debe ser válido y debe apuntar a un archivo abierto exitosamente por **bzopen()**.

longitud

Si no se especifica, **bzread()** leerá 1024 bytes (sin comprimir) a la vez.

Valores retornados

Regresa los datos ya sin compresión, o **FALSE** en caso de error.

Ejemplos

Ejemplo 1. Ejemplo de bzread()

```
<?php

$file = "/tmp/foo.bz2";
$bz = bzopen($file, "r") or die("Couldn't open $file");

$decompressed_file = '';
while (!feof($bz)) {
    $decompressed_file .= bzread($bz, 4096);
}
bzclose($bz);

echo "The contents of $file are: <br />\n";
echo $decompressed_file;

?>
```

Ver también

bzwrite()
feof()
bzopen()

bzwrite

(PHP 4 >= 4.0.4, PHP 5)

bzwrite -- Escritura segura de archivo bzip2 binario

Descripción

int **bzwrite** (resource bz, string datos [, int longitud])

bzwrite() escribe una cadena en el archivo bzip2 dado.

Lista de parámetros

bz

El apuntador de archivo. Este debe ser válido y debe apuntar a un archivo abierto exitosamente con **bzopen()**.

datos

Los datos escritos.

longitud

Si se da, la escritura se detendrá después de escribir *longitud* de bytes (sin comprimir) o se encuentra el fin de los *data* es alcanzado, lo que ocurra primero.

Valores retornados

Regresa el número de bytes escritos, o **FALSE** en caso de error.

Ejemplos

Ejemplo 1. Ejemplo de bzwrite()

```
<?php
$str = "uncompressed data";
$bz = bzopen("/tmp/foo.bz2", "w");
bzwrite($bz, $str, strlen($str));
bzclose($bz);
?>
```

Ver también

bzread()
bzopen()

VIII. Funciones de calendario

Introducción

La extensión calendar pone a disposición una serie de funciones para simplificar la conversión entre los distintos formatos de calendario. El intermediario ó estándar en que se basa es la Cuenta de Días Juliana. La Cuenta de Días Juliana es una cuenta que comienza mucho antes que lo que mucha gente podría necesitar contar (como alrededor del 4000 AC). Para convertir entre sistemas de calendario, primero deberá convertir a la Cuenta de Días Juliana y luego al sistema de su elección. ¡La Cuenta de Días es muy diferente del Calendario Juliano! Para más información sobre la Cuenta de Días Juliana visitar http://www.hermetic.ch/cal_stud/jdn.htm. Para más información sobre sistemas de calendario, visitar http://www.boogle.com/info/cal-overview.html. En estas instrucciones se han incluído extractos entrecomillados de dicha página.

Instalación

Para tener trabajando estas funciones, tiene que compilar PHP con *--enable-calendar*.

La versión para Windows de *PHP* tiene soporte nativo para esta extensión. No se necesita cargar ninguna extensión adicional para usar estas funciones.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

`CAL_GREGORIAN` (entero)

`CAL_JULIAN` (entero)

`CAL_JEWISH` (entero)

`CAL_FRENCH` (entero)

`CAL_NUM_CALS` (entero)

`CAL_DOW_DAYNO` (entero)

`CAL_DOW_SHORT` (entero)

`CAL_DOW_LONG` (entero)

`CAL_MONTH_GREGORIAN_SHORT` (entero)

`CAL_MONTH_GREGORIAN_LONG` (entero)

`CAL_MONTH_JULIAN_SHORT` (entero)

`CAL_MONTH_JULIAN_LONG` (entero)

`CAL_MONTH_JEWISH` (entero)

CAL_MONTH_FRENCH (entero)

Las siguientes constantes se pueden utilizar desde *PHP* 4.3.0 :

CAL_EASTER_DEFAULT (entero)

CAL_EASTER_ROMAN (entero)

CAL_EASTER_ALWAYS_GREGORIAN (entero)

CAL_EASTER_ALWAYS_JULIAN (entero)

Estas constantes están disponibles a partir de *PHP* 5.0.0 :

CAL_JEWISH_ADD_ALAFIM_GERESH (entero)

CAL_JEWISH_ADD_ALAFIM (entero)

CAL_JEWISH_ADD_GERESHAYIM (entero)

Tabla de contenidos

cal_days_in_month

(PHP 4 >= 4.1.0, PHP 5)

cal_days_in_month -- Devuelve el número de dias en un mes para un determinado año y calendario

Descripción

int **cal_days_in_month** (int calendario, int mes, int año)

Esta función devuelve el numero de dias en el *mes* del *año* para el calendario especificado *calendario*.

Ver también **jdtounix()**.

cal_from_jd

(PHP 4 >= 4.1.0, PHP 5)

cal_from_jd -- Convierte de Cuenta de Días Juliana a un calendario soportado.

Descripción

array **cal_from_jd** (int jd, int calendario)

cal_from_jd() convierte el dia de Cuenta Juliana definido en *jd* a un dia del *calendario* especificado. Los valores de *calendario* soportados son **CAL_GREGORIAN**, **CAL_JULIAN**, **CAL_JEWISH** and **CAL_FRENCH**.

Ver también **cal_to_jd()**.

cal_info

(PHP 4 >= 4.1.0, PHP 5)

cal_info -- Devuelve información sobre un calendario en particular.

Descripción

array **cal_info** ([int calendario])

cal_info() devuelve información sobre el *calendario* especificado ó sobre todos los calendarios soportados si no se especifica el parametro *calendario*.

La información del calendario es devuelta en una matriz conteniendo los elementos *calname*, *calsymbol*, *month*, *abbrevmonth* y *maxdaysinmonth*.

Si no se especifica *calendario*, se devuelve una matriz con informacion sobre todos los calendarios soportados. Esta funcionalidad estará disponible en PHP 5.

cal_to_jd

(PHP 4 >= 4.1.0, PHP 5)

cal_to_jd -- Convierte un calendario soportado a Cuenta de Días Juliana.

Descripción

int **cal_to_jd** (int calendario, int mes, int dia, int año)

cal_to_jd() Calcula la Cuenta de Días Juliana para un dia en el calendario especificado por *calendar*. Los *calendarios* soportados son **CAL_GREGORIAN**, **CAL_JULIAN**, **CAL_JEWISH** y

CAL_FRENCH.

Ver también **cal_to_jd()**.

easter_date

(PHP 3>= 3.0.9, PHP 4 , PHP 5)

easter_date -- devuelve la marca de tiempo UNIX para la medianoche de Pascua de un año dado

Descripción

int **easter_date** ([int anno])

Devuelve la marca de tiempo UNIX que corresponde a la medianoche de Pascua del año dado.

A partir de *PHP* 4.3.0, el parametro *anno* es opcional y si se omite, usa por defecto el año en curso según "localtime".

Aviso: Esta función generará un aviso si el año está fuera del rango para las marcas de tiempo del UNIX (es decir, antes de 1970 o después del 2037).

Ejemplo 1. ejemplo de easter_date()

```
echo date ("M-d-Y", easter_date(1999));        /* "Apr-04-1999" */
echo date ("M-d-Y", easter_date(2000));        /* "Apr-23-2000" */
echo date ("M-d-Y", easter_date(2001));        /* "Apr-15-2001" */
```

La fecha del Día de Pascua fue definida por el Concilio de Nicea en el 325 D.C. como el domingo tras la primera luna llena que cayera en ó después del equinoccio de Primavera. El equinoccio se supone que siempre cae en el 21 de marzo, de modo que el cálculo se reduce a determinar la fecha de la luna llena y la del domingo siguiente. El algoritmo usado aquí fue introducido en el año 532 por Dionisio Exiguo. Bajo el Calendario Juliano (para años anteriores al 1753), se usa un ciclo simple de 19 años para calcular las fases de la luna. Bajo el Calendario Gregoriano (años posteriores al 1753, diseñado por Clavio y Lilio, e introducido por el Papa Gregorio XIII en Octubre de 1582, y en Gran Bretaña y sus colonias en septiembre de 1752) se añaden dos factores de corrección para hacer el ciclo más preciso.

(El código se basa en un programa en C de Simon Kershaw, <webmaster@ely.anglican.org>)

Ver **easter_days()** para calcular la Pascua antes del 1970 o después del 2037.

easter_days

(PHP 3>= 3.0.9, PHP 4 , PHP 5)

easter_days -- Obtiene el número de días tras el 21 de marzo en que cae la Pascua en un año dado

Descripción

int **easter_days** ([int anno [, int metodo]])

Devuelve el número de días tras el 21 de marzo en que cae la Pascua en un año dado. Si no se especifica año, se asume el actual.

A partir de *PHP* 4.3.0, el parametro *anno* es opcional y si se omite, usa por defecto el año en curso según "localtime".

El parámetro *metodo* fue introducido en la version *PHP* 4.3.0 y permite calcular fechas de pascua basadas en el Calendario Gregoriano durante los años 1582 - 1752 si se le da el valor **CAL_EASTER_ROMAN**. Ver las constantes de calendario para más información sobre estas constantes.

Esta función se puede usar en lugar de **easter_date()** para calcular la Pascua para años que se salen del rango de las marcas de fecha del UNIX (o sea, antes del 1970 o después del 2037).

Ejemplo 1. ejemplo de easter_date()

```
echo easter_days (1999);        /* 14, i.e. April 4   */
echo easter_days (1492);        /* 32, i.e. April 22  */
echo easter_days (1913);        /*  2, i.e. March 23  */
```

La fecha del Día de Pascua fue definida por el Concilio de Nicea en el 325 D.C. como el domingo tras la primera luna llena que cayera en o después del equinoccio de Primavera. El equinoccio se supone que siempre cae en el 21 de marzo, de modo que el cálculo se reduce a determinar la fecha de la luna llena y la del domingo siguiente. El algoritmo usado aquí fue introducido en el año 532 por Dionisio Exiguo. Bajo el Calendario Juliano (para años anteriores al 1753), se usa un ciclo simple de 19 años para calcular las fases de la luna. Bajo el Calendario Gregoriano (años posteriores al 1753, diseñado por Clavio y Lilio, e introducido por el Papa Gregorio XIII en Octubre de 1582, y en Gran Bretaña y sus colonias en septiembre de 1752) se añaden dos factores de corrección para hacer el ciclo más preciso.

(El código se basa en un programa en C de Simon Kershaw, <webmaster@ely.anglican.org>)

Vea también **easter_date()**.

FrenchToJD

(PHP 3, PHP 4 , PHP 5)

FrenchToJD -- Convierte del Calendario Republicano Francés a la Cuenta de Días Juliana

Descripción

int **frenchtojd** (int mes, int dia, int anno)

Convierte una fecha del Calendario Republicano Francés a la Cuenta de Días Juliana.

Estas rutinas sólo convierten fechas entre los años 1 y 14 (fechas Gregorianas del 22 de septiembre de 1792 al 22 de septiembre de 1806). Esto cubre ampliamente el periodo en el que estuvo en uso este calendario.

GregorianToJD

(PHP 3, PHP 4 , PHP 5)

GregorianToJD -- Convierte de fecha Gregoriana a la Cuenta de Días Juliana

Descripción

int **gregoriantojd** (int mes, int dia, int anno)

El rango válido para el Calendario Gregoriano es desde el 4714 A.C. hasta el 9999 D.C.

Aunque este programa puede manejar fechas tan lejanas como el 4714 A.C., usarlo no tendría sentido. El calendario Gregoriano fue instituído el 15 de octubre de 1582 (o el 5 de octubre de 1582 en el calendario Juliano). Algunos países no lo aceptaron hasta mucho después. Por ejemplo, Gran Bretaña se convirtió en 1752, la URSS en 1918 y Grecia en 1923. Muchos países europeos usaron el calendario Juliano antes que el Gregoriano.

Ejemplo 1. Funciones de calendario

```
<?php
$jd = GregorianToJD (10,11,1970);
echo "$jd\n";
$gregorian = JDToGregorian ($jd);
echo "$gregorian\n";
?>
```

JDDayOfWeek

(PHP 3, PHP 4 , PHP 5)

JDDayOfWeek -- Devuelve el día de la semana

Descripción

mixed **jddayofweek** (int diajuliano, int modo)

Devuelve el día de la semana. Dependiendo del modo, devuelve un entero ó una cadena.

Tabla 1. Modos para el día de la semana

Modo	Significado
0	devuelve el día de la semana como entero (0=domingo, 1=lunes, etc)
1	devuelve una cadena con el día de la semana (inglés, gregoriano)
2	devuelve una cadena con el día de la semana abreviado (inglés, gregoriano)

JDMonthName

(PHP 3, PHP 4 , PHP 5)

JDMonthName -- Devuelve el nombre de un mes

Descripción

string **jdmonthname** (int diajuliano, int modo)

Devuelve una cadena que contiene el nombre del mes. *modo* le dice a esta función a qué calendario

debe convertir la Cuenta de Días Juliana, y qué tipo de nombres de mes debe devolver.

Tabla 1. Modos de calendario

Modo	Significado	Valores
0	Gregoriano - abreviado	Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec
1	Gregoriano	January, February, March, April, May, June, July, August, September, October, Novemb er, December
2	Juliano - abreviado	Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec
3	Juliano	January, February, March, April, May, June, July, August, September, October, Novemb er, December
4	Judío	Tishri, Heshvan, Kislev, Tevet, Shevat, AdarI, AdarII, Nisan, Iyyar, Sivan, Tammuz, Av, Elul
5	Republicano Francés	Vendemiaire, Brumaire, Frimaire, Nivose, Pluviose, Ventose, Germinal, Floreal, Prair ial, Messidor, Thermidor, Fructidor, Extra

JDToFrench

(PHP 3, PHP 4 , PHP 5)

JDToFrench -- Convierte de Cuenta de Días al Calendario Republicano Francés

Descripción

string **jdtofrench** (int diajuliano)

Convierte una Cuenta de Días Juliana al Calendario Republicano Francés.

JDToGregorian

(PHP 3, PHP 4 , PHP 5)

JDToGregorian -- Convierte de Cuenta de Días a fecha Gregoriana

Descripción

string **jdtogregorian** (int diajuliano)

Convierte de Cuenta de Días Juliana a una cadena que contiene la fecha Gregoriana en formato "mes/día/año"

jdtojewish

(PHP 3, PHP 4 , PHP 5)

jdtojewish -- Convierte de cuenta de días juliana a calendario judío

Descripción

string **jdtojewish** (int diajuliano [, bool hebreo [, int fl]])

Convierte una cuenta de días juliana al calendario judío.

Los parámetros opcionales *hebreo* y *fl* está disponibles a partir de PHP 5.0.0

Si al parámetro *hebreo* se le asigna el valor **TRUE**, el parámetro *fl* se usa para el formato de salida hebreo. Los formatos disponibles son: **CAL_JEWISH_ADD_ALAFIM_GERESH**, **CAL_JEWISH_ADD_ALAFIM**, **CAL_JEWISH_ADD_GERESHAYIM**.

Ejemplo 1. Ejemplo jdtojewish()

```
<?php
echo jdtojewish(gregoriantojd(10,8,2002), true,
       CAL_JEWISH_ADD_GERESHAYIM + CAL_JEWISH_ADD_ALAFIM + CAL_JEWISH_ADD_ALAFIM_GERESH
?>
```

JDToJulian

(PHP 3, PHP 4 , PHP 5)

JDToJulian -- Convierte de Cuenta de Días Juliana a Calendario Juliano

Descripción

string **jdtojulian** (int diajuliano)

Convierte una Cuenta de Días Juliana a una cadena que contiene la fecha del Calendario Juliano en formato "mes/día/año".

jdtounix

(PHP 4 , PHP 5)

jdtounix -- Convierte un dia Juliano a UNIX timestamp

Descripción

int **jdtounix** (int jday)

Esta funció devuelve el "UNIX timestamp" correspondiante a el dia Juliano definido en *jday* ó falso (**FALSE**) si *jday* no se encuentra en la época UNIX (años entre 1970 y 2037 ó 2440588 <= *jday* <= 2465342). El tiempo devuelto es localtime (y no GMT).

Ver también **unixtojd()**.

JewishToJD

(PHP 3, PHP 4 , PHP 5)

JewishToJD -- Convierte del Calendario Judío a la Cuenta de Días Juliana

Descripción

int **jewishtojd** (int mes, int dia, int anno)

Aunque este programa puede manejar fechas tan lejanas como el año 1 (3761 A.C.), usarlo no tendría sentido. El Calendario Judío ha estado en uso miles de años, pero en los días primeros no había una fórmula que calculara el comienzo de un mes. Un mes comenzaba cuando se veía por primera vez la luna nueva.

JulianToJD

(PHP 3, PHP 4 , PHP 5)

JulianToJD -- Convierte de Calendario Juliano a Cuenta de Días Juliana

Descripción

int **juliantojd** (int mes, int dia, int anno)

Rango válido para el Calendario Juliano: del 4713 A.C al 9999 D.C.

Aunque este programa puede manejar fechas tan lejanas como el 4713 A.C., usarlo no tendría sentido. El calendario se creó en el 46 A.C., pero sus detalles no se estabilizaron hasta al menos el 8 D.C., y quizás no lo hiciera hasta el siglo IV. Además, el comienzo de un año variaba de una a otra cultura: no todas aceptaban enero como el primer mes.

Atención
Recordar que el actual sistema de calendario en uso en todo el mundo es el calendario Gregoriano. **gregoriantojd()** puede ser usada para convertir los dias del calendario Gregoriano a Cuenta de Días Juliana.

unixtojd

(PHP 4 , PHP 5)

unixtojd -- Convierte de UNIX timestamp a dia Juliano

Descripción

int **unixtojd** ([int timestamp])

Devuelve el dia Juliano correspondiente a un UNIX *timestamp* (segundos desde 01.01.1970), ó al dia actual si no se especifica *timestamp*

Ver tambien**jdtounix()**.

IX. Funciones del API de CCVS

Introducción

Estas funciones interaccionan con el API de CCVS, permitiendo trabajar con CCVS directamente desde un script PHP. CCVS es la solución de RedHat para el intermediario en el procesamiento de tarjetas de crédito. Permite conectar directamente con las centrales de las tarjetas desde una máquina *nix con un módem. Usando el módulo para PHP de CCVS, se pueden procesar tarjetas de crédito directamente desde vuestros scripts en PHP.

> **Nota:** CCVS ha sido discontinuado por Red Hat y no existen planes de ofrecer nuevas funcionalidades ó contratos de ayuda. Los que necesiten usar esta funcionalidad pueden probar MCVE by Main Street Softworks. Es similar en diseño y tiene documentación para su uso con PHP.
>
> Esta extensión ha sido removida de *PHP* y no se encuentra disponible a partir de PHP 4.3.0. Si quereis utilizar la capacidad de procesar tarjetas de crédito, usar MCVE.

Instalación

To enable CCVS Support in PHP, first verify your CCVS installation directory. You will then need to configure PHP with the *--with-ccvs* option. If you use this option without specifying the path to your CCVS installation, PHP will attempt to look in the default CCVS Install location (`/usr/local/ccvs`). If CCVS is in a non-standard location, run configure with: *--with-ccvs=[DIR]*, where DIR is the path to your CCVS installation. Please note that CCVS support requires that DIR/lib and DIR/include exist, and include `cv_api.h` under the include directory and `libccvs.a` under the lib directory.

Additionally, a ccvsd process will need to be running for the configurations you intend to use in your PHP scripts. You will also need to make sure the PHP Processes are running under the same user as your CCVS was installed as (e.g. if you installed CCVS as user 'ccvs', your PHP processes must run as 'ccvs' as well.)

Ver también

RedHat ha dejado de soportar CCVS; De todas maneras un manual un poco anticuado está todavía disponible en http://redhat.com/docs/manuals/ccvs/.

Tabla de contenidos

ccvs_lookup -- Busca un item de un tipo en particular en la base de datos #
ccvs_new -- Crea una nueva, transacción en blanco
ccvs_report -- Devuelve el estado del proceso de comunicación en background
ccvs_return -- Transfiere fondos del comerciante al titular de la tarjeta
ccvs_reverse -- Realiza una revocación completa en una autorización ya procesada
ccvs_sale -- Transfiere fondos del titular de la tarjeta al comerciante
ccvs_status -- Chequear el estado de una factura
ccvs_textvalue -- Obtiene el valor de retorno de texto para una llamada anterior a una función
ccvs_void -- Realizar una revocación completa en una transacción completada

ccvs_add

(4.0.2 - 4.2.3 only)

ccvs_add -- Añadir datos a una transacción

Descripción

cadena **ccvs_add** (cadena sesión, cadena factura, cadena argtype, cadena argval)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ccvs_auth

(4.0.2 - 4.2.3 only)

ccvs_auth -- Realiza un test de una autorización a crédito en una transacción

Descripcion

cadena **ccvs_auth** (cadena sesión, cadena factura)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ccvs_command

(4.0.2 - 4.2.3 only)

ccvs_command -- Ejecuta un comando que es peculiar para un protocolo concreto, y que no está disponible en el API general de CCVS

Descripcion

cadena **ccvs_command** (cadena sesión, cadena tipo, cadena argval)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ccvs_count

(4.0.2 - 4.2.3 only)

ccvs_count -- Encuentra cuantas transacciones de un tipo dado están almacenadas en el sistema

Descripción

entero **ccvs_count** (cadena sesión, cadena tipo)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ccvs_delete

(4.0.2 - 4.2.3 only)

ccvs_delete -- Borra una transacción

Descripción

cadena **ccvs_delete** (cadena sesión, cadena factura)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ccvs_done

(4.0.2 - 4.2.3 only)

ccvs_done -- Finaliza el motor de CCVS y hace una limpieza

Descripcion

cadena **ccvs_done** (cadena sesió)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ccvs_init

(4.0.2 - 4.2.3 only)

ccvs_init -- Inicializa un CCVS para usarlo

Descripción

cadena **ccvs_init** (cadena nombre)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ccvs_lookup

(4.0.2 - 4.2.3 only)

ccvs_lookup -- Busca un item de un tipo en particular en la base de datos #

Descripción

cadena **ccvs_lookup** (cadena sesión, cadena factura, entero inum)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ccvs_new

(4.0.2 - 4.2.3 only)

ccvs_new -- Crea una nueva, transacción en blanco

Descripcion

cadena **ccvs_new** (cadena sesión, cadena cadena)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ccvs_report

(4.0.2 - 4.2.3 only)

ccvs_report -- Devuelve el estado del proceso de comunicación en background

Descripcion

cadena **ccvs_report** (cadena sesión, cadena tipo)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ccvs_return

(4.0.2 - 4.2.3 only)

ccvs_return -- Transfiere fondos del comerciante al titular de la tarjeta

Descripción

cadena **ccvs_return** (cadena sesión, cadena factura)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ccvs_reverse

(4.0.2 - 4.2.3 only)

ccvs_reverse -- Realiza una revocación completa en una autorización ya procesada

Descripcion

cadena **ccvs_reverse** (cadena sesión, cadena factura)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ccvs_sale

(4.0.2 - 4.2.3 only)

ccvs_sale -- Transfiere fondos del titular de la tarjeta al comerciante

Descripción

cadena **ccvs_sale** (cadena sesión, cadena factura)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ccvs_status

(4.0.2 - 4.2.3 only)

ccvs_status -- Chequear el estado de una factura

Descripción

cadena **ccvs_status** (cadena sesión, cadena factura)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ccvs_textvalue

(4.0.2 - 4.2.3 only)

ccvs_textvalue -- Obtiene el valor de retorno de texto para una llamada anterior a una función

Descripción

cadena **ccvs_textvalue** (cadena sesión)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ccvs_void

(4.0.2 - 4.2.3 only)

ccvs_void -- Realizar una revocación completa en una transacción completada

Descripción

cadena **ccvs_void** (cadena sesión, cadena factura)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

X. Classkit Functions

Introducción

These functions allow the dynamic manipulation of PHP classes, at runtime.

Instalación

Esta extension PECL no esta ligada a PHP.

Mas informacion sobre nuevos lanzamientos, descargas ficheros de fuentes, informacion sobre los responsables asi como un 'CHANGELOG', se puede encontrar aqui: http://pecl.php.net/package/classkit.

Podeis descargar esta DLL de las extensiones PECL desde la pagina PHP Downloads o desde http://snaps.php.net/.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

`CLASSKIT_ACC_PRIVATE` (**int**)

Marks the method *private*

`CLASSKIT_ACC_PROTECTED` (**int**)

Marks the method *protected*

`CLASSKIT_ACC_PUBLIC` (**int**)

Marks the method *public*

Tabla de contenidos

classkit_import

(no version information, might be only in CVS)

classkit_import -- Import new class method definitions from a file

Descripción

array **classkit_import** (string filename)

> **Nota:** Esta función no puede usarse para manipular el método actualmente en ejecución

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Lista de parámetros

filename

> The filename of the class method definitions to import

Valores retornados

Associative array of imported methods

Ejemplos

Ejemplo 1. classkit_import() example

```
<?php
// file: newclass.php
class Example {
    function foo() {
        return "bar!\n";
    }
}
?>
```

```
<?php
// requires newclass.php (see above)
class Example {
    function foo() {
        return "foo!\n";
    }
}

$e = new Example();

// output original
echo $e->foo();

// import replacement method
classkit_import('newclass.php');

// output imported
echo $e->foo();

?>
```

El resultado del ejemplo seria:

```
foo!
bar!
```

Ver también

classkit_method_add()
classkit_method_copy()

classkit_method_add

(no version information, might be only in CVS)

classkit_method_add -- Dynamically adds a new method to a given class

Descripción

bool **classkit_method_add** (string classname, string methodname, string args, string code [, int flags])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Lista de parámetros

classname

The class to which this method will be added

methodname

The name of the method to add

args

Comma-delimited list of arguments for the newly-created method

code

The code to be evaluated when *methodname* is called

flags

The type of method to create, can be **CLASSKIT_ACC_PUBLIC**, **CLASSKIT_ACC_PROTECTED** or **CLASSKIT_ACC_PRIVATE**

Nota: This parameter is only used as of PHP 5, because, prior to this, all methods were public.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplos

Ejemplo 1. classkit_method_add() example

```
<?php
class Example {
    function foo() {
        echo "foo!\n";
    }
}

// create an Example object
$e = new Example();

// Add a new public method
classkit_method_add(
    'Example',
    'add',
    '$num1, $num2',
    'return $num1 + $num2;',
    CLASSKIT_ACC_PUBLIC
);

// add 12 + 4
echo $e->add(12, 4);
?>
```

El resultado del ejemplo seria:

```
16
```

Ver también

classkit_method_copy()
classkit_method_redefine()
classkit_method_remove()
classkit_method_rename()
create_function()

classkit_method_copy

(no version information, might be only in CVS)

classkit_method_copy -- Copies a method from class to another

Descripción

bool **classkit_method_copy** (string dClass, string dMethod, string sClass [, string sMethod])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Lista de parámetros

dClass

Destination class for copied method

dMethod

Destination method name

sClass

Source class of the method to copy

sMethod

Name of the method to copy from the source class. If this parameter is omitted, the value of *dMethod* is assumed.

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Ejemplos

Ejemplo 1. classkit_method_copy() example

```
<?php
class Foo {
    function example() {
        return "foo!\n";
    }
}

class Bar {
    // initially, no methods
}

// copy the example() method from the Foo class to the Bar class, as baz()
classkit_method_copy('Bar', 'baz', 'Foo', 'example');

// output copied function
echo Bar::baz();
?>
```

El resultado del ejemplo seria:

```
foo!
```

Ver también

classkit_method_add()
classkit_method_redefine()
classkit_method_remove()
classkit_method_rename()

classkit_method_redefine

(no version information, might be only in CVS)

classkit_method_redefine -- Dynamically changes the code of the given method

Descripción

bool **classkit_method_redefine** (string classname, string methodname, string args, string code [, int flags])

> **Nota:** Esta función no puede usarse para manipular el método actualmente en ejecución

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Lista de parámetros

classname

> The class in which to redefine the method

methodname

The name of the method to redefine

args

Comma-delimited list of arguments for the redefined method

code

The new code to be evaluated when *methodname* is called

flags

The redefined method can be **CLASSKIT_ACC_PUBLIC**, **CLASSKIT_ACC_PROTECTED** or **CLASSKIT_ACC_PRIVATE**

Nota: This parameter is only used as of PHP 5, because, prior to this, all methods were public.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplos

Ejemplo 1. classkit_method_redefine() example

```
<?php
class Example {
    function foo() {
        return "foo!\n";
    }
}

// create an Example object
$e = new Example();

// output Example::foo() (before redefine)
echo "Before: " . $e->foo();

// Redefine the 'foo' method
classkit_method_redefine(
    'Example',
    'foo',
    '',
    'return "bar!\n";',
    CLASSKIT_ACC_PUBLIC
);

// output Example::foo() (after redefine)
echo "After: " . $e->foo();
?>
```

El resultado del ejemplo seria:

```
Before: foo!
After: bar!
```

Ver también

classkit_method_add()

classkit_method_copy()
classkit_method_remove()
classkit_method_rename()

classkit_method_remove

(no version information, might be only in CVS)

classkit_method_remove -- Dynamically removes the given method

Descripción

bool **classkit_method_remove** (string classname, string methodname)

> **Nota:** Esta función no puede usarse para manipular el método actualmente en ejecución

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Lista de parámetros

classname

> The class in which to remove the method

methodname

> The name of the method to remove

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Ejemplos

Ejemplo 1. classkit_method_remove() example

```php
<?php
class Example {
    function foo() {
        return "foo!\n";
    }

    function bar() {
        return "bar!\n";
    }
}

// Remove the 'foo' method
classkit_method_remove(
    'Example',
    'foo'
);

echo implode(' ', get_class_methods('Example'));

?>
```

El resultado del ejemplo seria:

```
bar
```

Ver también

classkit_method_add()
classkit_method_copy()
classkit_method_redefine()
classkit_method_rename()

classkit_method_rename

(no version information, might be only in CVS)

classkit_method_rename -- Dynamically changes the name of the given method

Descripción

bool **classkit_method_rename** (string classname, string methodname, string newname)

Nota: Esta función no puede usarse para manipular el método actualmente en ejecución

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Lista de parámetros

classname

The class in which to rename the method

methodname

The name of the method to rename

newname

The new name to give to the renamed method

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplos

Ejemplo 1. classkit_method_rename() example

```
<?php
class Example {
    function foo() {
        return "foo!\n";
    }
}

// Rename the 'foo' method to 'bar'
classkit_method_rename(
    'Example',
    'foo',
    'bar'
);

// output renamed function
echo Example::bar();
?>
```

El resultado del ejemplo seria:

```
foo!
```

Ver también

classkit_method_add()
classkit_method_copy()
classkit_method_redefine()
classkit_method_remove()

XI. Funciones de Clases/Objetos

Introducción

Estas funciones permiten obtener información sobre clases y objetos. Se puede obtener el nombre de la clase a la que pertenece un objeto, asi como las propiedades de sus miembros y métodos. Usando estas funciones se puede obtener no solo lo comentado en la frase anterior, también se puede obtener la familia del objeto (p.ej. qué clase está extendiendo la clase a la que pertenece el objeto).

Requirimientos

No se necesitan bibliotecas externas para construir esta extensión

---

Instalación

No se necesita ninguna instalación para usar estas funciones, son parte del núcleo de PHP.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Esta extensión no tiene ninguna constante definida.

Ejemplos

En este ejemplo, definimos primero una clase base y una extensión de esta clase. La clase base define un vegetal genérico, si es comestible y su color. La subclase *Spinach* añade un metodo para cocinarlo y otro para saber si esta cocinado.

Ejemplo 1. classes.inc

```php
<?php

// base class with member properties and methods
class Vegetable {

    var $edible;
    var $color;

    function Vegetable( $edible, $color="green" ) {
        $this->edible = $edible;
        $this->color = $color;
    }

    function is_edible() {
        return $this->edible;
    }

    function what_color() {
        return $this->color;
    }

} // end of class Vegetable

// extends the base class
class Spinach extends Vegetable {

    var $cooked = false;

    function Spinach() {
        $this->Vegetable( true, "green" );
    }

    function cook_it() {
        $this->cooked = true;
    }

    function is_cooked() {
        return $this->cooked;
    }

} // end of class Spinach

?>
```

Creamos 2 objetos de estas clases e imprimimos información sobre ellos, incluyendo la jerarquia de clases a la que pertenecen. También definimos algunas funciones, especialmente para imprimir las variables de una manera ordenada.

Ejemplo 2. test_script.php

```
<pre>
<?php

include "classes.inc";

// utility functions

function print_vars($obj) {
    $arr = get_object_vars($obj);
    while (list($prop, $val) = each($arr))
        echo "\t$prop = $val\n";
}

function print_methods($obj) {
    $arr = get_class_methods(get_class($obj));
    foreach ($arr as $method)
        echo "\tfunction $method()\n";
}

function class_parentage($obj, $class) {
    global $$obj;
    if (is_subclass_of($$obj, $class)) {
        echo "Object $obj belongs to class ".get_class($$obj);
        echo " a subclass of $class\n";
    } else {
        echo "Object $obj does not belong to a subclass of $class\n";
    }
}

// instantiate 2 objects

$veggie = new Vegetable(true,"blue");
$leafy = new Spinach();

// print out information about objects
echo "veggie: CLASS ".get_class($veggie)."\n";
echo "leafy: CLASS ".get_class($leafy);
echo ", PARENT ".get_parent_class($leafy)."\n";

// show veggie properties
echo "\nveggie: Properties\n";
print_vars($veggie);

// and leafy methods
echo "\nleafy: Methods\n";
print_methods($leafy);

echo "\nParentage:\n";
class_parentage("leafy", "Spinach");
class_parentage("leafy", "Vegetable");
?>
</pre>
```

One important thing to note in the example above is that the object *$leafy* is an instance of the class **Spinach** which is a subclass of **Vegetable**, therefore the last part of the script above will output:

```
[...]
Parentage:
Object leafy does not belong to a subclass of Spinach
Object leafy belongs to class spinach a subclass of Vegetable
```

Tabla de contenidos

get_class -- Returns the name of the class of an object
get_declared_classes -- Returns an array with the name of the defined classes
get_declared_interfaces -- Returns an array of all declared interfaces
get_object_vars -- Devuelve un vector de propiedades del objecto
get_parent_class -- Retrieves the parent class name for object or class
interface_exists -- Checks if the interface has been defined
is_a -- Returns **TRUE** if the object is of this class or has this class as one of its parents
is_subclass_of -- Returns **TRUE** if the object has this class as one of its parents
method_exists -- Comprueba que el metódo de clase existe

call_user_method_array

(PHP 4 >= 4.0.5, PHP 5)

call_user_method_array -- Call a user method given with an array of parameters [deprecated]

Description

mixed **call_user_method_array** (string method_name, object &obj, array paramarr)

Aviso
The **call_user_method_array()** function is deprecated as of PHP 4.1.0, use the **call_user_func_array()** variety with the *array(&$obj, "method_name")* syntax instead.

Calls the method referred by *method_name* from the user defined *obj* object, using the parameters in *paramarr*.

See also: **call_user_func_array()**, and **call_user_func()**.

call_user_method

(PHP 3>= 3.0.3, PHP 4 , PHP 5)

call_user_method -- Call a user method on an specific object [deprecated]

Description

mixed **call_user_method** (string method_name, object &obj [, mixed parameter [, mixed ...]])

Aviso
The **call_user_method()** function is deprecated as of PHP 4.1.0, use the **call_user_func()** variety with the *array(&$obj, "method_name")* syntax instead.

Calls the method referred by *method_name* from the user defined *obj* object. An example of usage is below, where we define a class, instantiate an object and use **call_user_method()** to call indirectly its *print_info* method.

```php
<?php
class Country {
    var $NAME;
    var $TLD;

    function Country($name, $tld)
    {
        $this->NAME = $name;
        $this->TLD = $tld;
    }

    function print_info($prestr = "")
    {
        echo $prestr . "Country: " . $this->NAME . "\n";
        echo $prestr . "Top Level Domain: " . $this->TLD . "\n";
    }
}

$cntry = new Country("Peru", "pe");

echo "* Calling the object method directly\n";
$cntry->print_info();

echo "\n* Calling the same method indirectly\n";
call_user_method("print_info", $cntry, "\t");
?>
```

See also **call_user_func_array()**, and **call_user_func()**.

class_exists

(PHP 4 , PHP 5)

class_exists -- Checks if the class has been defined

Description

bool **class_exists** (string class_name [, bool autoload])

This function returns **TRUE** if the class given by *class_name* has been defined, **FALSE** otherwise.

Ejemplo 1. class_exists() example

```php
<?php
// Check the class exists before trying to use it
if (class_exists('MyClass')) {
    $myclass = new MyClass();
}

?>
```

class_exists() will attempt to call __autoload by default, if you don't want **class_exists()** to call __autoload, you can set the parameter *autoload* to **FALSE**.

Ejemplo 2. *autoload* parameter example

```
<?php
function __autoload($class)
{
    include($class . '.php');

    // Check to see if the include declared the class
    if (!class_exists($class, false)) {
        trigger_error("Unable to load class: $class", E_USER_WARNING);
    }
}

if (class_exists('MyClass')) {
    $myclass = new MyClass();
}

?>
```

Nota: The *autoload* parameter was added in PHP 5

See also **interface_exists()**, and **get_declared_classes()**.

get_class_methods

(PHP 4 , PHP 5)

get_class_methods -- Devuelve un vector (matriz unidimensional) con los nombres de los métodos de la clase en question.

Descripción

vector **get_class_methods** (string class_name)

Esta función devuelve un vector con los nombres de los métodos definidos en la clase especificada como *class_name*.

Nota: A partir de PHP 4.0.6, se puede especificar el objeto a sí mismo en vez de *class_name*. Por ejemplo:

```
$class_methods = get_class_methods($my_class); // see below the full example
```

Ejemplo 1. get_class_methods() ejemplo

```php
<?php

class myclass {
    // constructor
    function myclass() {
        return(TRUE);
    }

    // method 1
    function myfunc1() {
        return(TRUE);
    }

    // method 2
    function myfunc2() {
        return(TRUE);
    }
}

$my_object = new myclass();

$class_methods = get_class_methods(get_class($my_object));

foreach ($class_methods as $method_name) {
    echo "$method_name\n";
}

?>
```

Producira:

```
myclass
myfunc1
myfunc2
```

Ver también **get_class_vars()** y **get_object_vars()**.

get_class_vars

(PHP 4 , PHP 5)

get_class_vars -- Devuelve una matriz con las propiedades (inicializadas por defecto) de la clase

Descripción

array **get_class_vars** (string class_name)

Esta función devuelve un vector con las propiedades que han sido inicializadas por defecto en la clase. Los elementos de este vector están estan organizados de la forma *varname => value*.

> **Nota:** Con anterioridad a PHP 4.2.0, las variables de la clase que no estén inicializadas, no será presentadas por **get_class_vars()**.

Ejemplo 1. get_class_vars() ejemplo

```
<?php

class myclass {

    var $var1; // this has no default value...
    var $var2 = "xyz";
    var $var3 = 100;

    // constructor
    function myclass() {
        return(TRUE);
    }
}

$my_class = new myclass();

$class_vars = get_class_vars(get_class($my_class));

foreach ($class_vars as $name => $value) {
    echo "$name : $value\n";
}

?>
```

Producira:

```
// Before PHP 4.2.0
var2 : xyz
var3 : 100

// As of PHP 4.2.0
var1 :
var2 : xyz
var3 : 100
```

Ver también **get_class_methods()**, **get_object_vars()**

get_class

(PHP 4 , PHP 5)

get_class -- Returns the name of the class of an object

Description

string **get_class** (object obj)

This function returns the name of the class of which the object *obj* is an instance. Returns **FALSE** if *obj* is not an object.

> **Nota:** A class defined in a PHP extension is returned in its original notation. In PHP 4 **get_class()** returns a user defined class name in lowercase, but in PHP 5 it will return the class name in it's original notation too, just like class names from PHP extensions.

Ejemplo 1. Using get_class()

```
<?php

class foo {
    function foo()
    {
    // implements some logic
    }

    function name()
    {
        echo "My name is " , get_class($this) , "\n";
    }
}

// create an object
$bar = new foo();

// external call
echo "Its name is " , get_class($bar) , "\n";

// internal call
$bar->name();

?>
```

El resultado del ejemplo seria:

```
Its name is foo
My name is foo
```

See also **get_parent_class()**, **gettype()**, and **is_subclass_of()**.

get_declared_classes

(PHP 4 , PHP 5)

get_declared_classes -- Returns an array with the name of the defined classes

Description

array **get_declared_classes** (void)

This function returns an array of the names of the declared classes in the current script.

> **Nota:** In PHP 4.0.1pl2, three extra classes are returned at the beginning of the array: **stdClass** (defined in `Zend/zend.c`), **OverloadedTestClass** (defined in `ext/standard/basic_functions.c`) and **Directory** (defined in `ext/standard/dir.c`).
>
> Also note that depending on what libraries you have compiled into PHP, additional classes could be present. This means that you will not be able to define your own classes using these names. There is a list of predefined classes in the Predefined Classes section of the appendices.

Ejemplo 1. get_declared_classes() example

```
<?php
print_r(get_declared_classes());
?>
```

El resultado del ejemplo seria algo similar a:

```
Array
(
    [0] => stdClass
    [1] => __PHP_Incomplete_Class
    [2] => Directory
)
```

See also **class_exists()**, and **get_declared_interfaces()**.

get_declared_interfaces

(PHP 5)

get_declared_interfaces -- Returns an array of all declared interfaces

Description

array **get_declared_interfaces** (void)

This function returns an array of the names of the declared interfaces in the current script.

Ejemplo 1. get_declared_interfaces() example

```
<?php
print_r(get_declared_interfaces());
?>
```

El resultado del ejemplo seria algo similar a:

```
Array
(
    [0] => Traversable
    [1] => IteratorAggregate
    [2] => Iterator
    [3] => ArrayAccess
    [4] => reflector
    [5] => RecursiveIterator
    [6] => SeekableIterator
)
```

See also **get_declared_classes()**.

get_object_vars

(PHP 4 , PHP 5)

get_object_vars -- Devuelve un vector de propiedades del objecto

Descripción

array **get_object_vars** (object obj)

Esta función devuelve una matriz con las propiedades definidas en el objecto especificado como *obj*.

> **Nota:** Con anterioridad a PHP 4.2.0, si las variables declaradas en la clase a la que pertenece *obj*, no les ha sido asignado un valor, no serán devueltas en el vector. En versiones posteriores a PHP 4.2.0 se les asignará un valor **NULL**.

Ejemplo 1. Uso de get_object_vars()

```
<?php
class Point2D {
    var $x, $y;
    var $label;

    function Point2D($x, $y) {
        $this->x = $x;
        $this->y = $y;
    }

    function setLabel($label) {
        $this->label = $label;
    }

    function getPoint() {
        return array("x" => $this->x,
                     "y" => $this->y,
                     "label" => $this->label);
    }
}

// "$label" is declared but not defined
$p1 = new Point2D(1.233, 3.445);
print_r(get_object_vars($p1));

$p1->setLabel("point #1");
print_r(get_object_vars($p1));

?>
```

El resultado de este programa es:

```
Array
 (
     [x] => 1.233
     [y] => 3.445
     [label] =>
 )

 Array
 (
     [x] => 1.233
     [y] => 3.445
     [label] => point #1
 )
```

Ver tambien **get_class_methods()** y **get_class_vars()**

get_parent_class

(PHP 4 , PHP 5)

get_parent_class -- Retrieves the parent class name for object or class

Description

string **get_parent_class** (mixed obj)

If *obj* is an object, returns the name of the parent class of the class of which *obj* is an instance.

If *obj* is a string, returns the name of the parent class of the class with that name. This functionality was added in PHP 4.0.5.

Ejemplo 1. Using get_parent_class()

```
<?php

class dad {
    function dad()
    {
    // implements some logic
    }
}

class child extends dad {
    function child()
    {
        echo "I'm " , get_parent_class($this) , "'s son\n";
    }
}

class child2 extends dad {
    function child2()
    {
        echo "I'm " , get_parent_class('child2') , "'s son too\n";
    }
}

$foo = new child();
$bar = new child2();

?>
```

El resultado del ejemplo seria:

```
I'm dad's son
I'm dad's son too
```

See also **get_class()** and **is_subclass_of()**.

interface_exists

(no version information, might be only in CVS)

interface_exists -- Checks if the interface has been defined

Description

bool **interface_exists** (string interface_name [, bool autoload])

This function returns **TRUE** if the interface given by *interface_name* has been defined, **FALSE** otherwise.

Ejemplo 1. interface_exists() example

```
<?php
// Check the interface exists before trying to use it
if (interface_exists('MyInterface')) {
    class MyClass implements MyInterface
    {
        // Methods
    }
}

?>
```

interface_exists() will attempt to call __autoload by default, if you don't want **interface_exists()** to call __autoload, you can set the parameter *autoload* to **FALSE**.

See also **class_exists()**.

is_a

(PHP 4 >= 4.2.0, PHP 5)

is_a -- Returns **TRUE** if the object is of this class or has this class as one of its parents

Description

bool **is_a** (object object, string class_name)

This function returns **TRUE** if the object is of this class or has this class as one of its parents, **FALSE** otherwise.

Ejemplo 1. is_a() example

```
<?php
// define a class
class WidgetFactory
{
  var $oink = 'moo';
}

// create a new object
$WF = new WidgetFactory();

if (is_a($WF, 'WidgetFactory')) {
  echo "yes, \$WF is still a WidgetFactory\n";
}
?>
```

The **is_a()** function is deprecated as of PHP 5 in favor of the instanceof type operator. In the above example we could use the following in PHP 5:

Ejemplo 2. Using the *instanceof* operator in PHP 5

```
<?php
if ($WF instanceof WidgetFactory) {
    echo 'Yes, $WF is a WidgetFactory';
}
?>
```

See also **get_class()**, **get_parent_class()**, and **is_subclass_of()**.

is_subclass_of

(PHP 4 , PHP 5)

is_subclass_of -- Returns **TRUE** if the object has this class as one of its parents

Description

bool **is_subclass_of** (mixed object, string class_name)

This function returns **TRUE** if the object *object*, belongs to a class which is a subclass of *class_name*, **FALSE** otherwise.

Nota: Since PHP 5.0.3 you may also specify the *object* parameter as a string (the name of the class).

Ejemplo 1. is_subclass_of() example

```
<?php
// define a class
class WidgetFactory
{
  var $oink = 'moo';
}

// define a child class
class WidgetFactory_Child extends WidgetFactory
{
  var $oink = 'oink';
}

// create a new object
$WF = new WidgetFactory();
$WFC = new WidgetFactory_Child();

if (is_subclass_of($WFC, 'WidgetFactory')) {
  echo "yes, \$WFC is a subclass of WidgetFactory\n";
} else {
  echo "no, \$WFC is not a subclass of WidgetFactory\n";
}


if (is_subclass_of($WF, 'WidgetFactory')) {
  echo "yes, \$WF is a subclass of WidgetFactory\n";
} else {
  echo "no, \$WF is not a subclass of WidgetFactory\n";
}


// usable only since PHP 5.0.3
if (is_subclass_of('WidgetFactory_Child', 'WidgetFactory')) {
  echo "yes, WidgetFactory_Child is a subclass of WidgetFactory\n";
} else {
  echo "no, WidgetFactory_Child is not a subclass of WidgetFactory\n";
}
?>
```

El resultado del ejemplo seria:

```
yes, $WFC is a subclass of WidgetFactory
no, $WF is not a subclass of WidgetFactory
yes, WidgetFactory_Child is a subclass of WidgetFactory
```

See also **get_class()**, **get_parent_class()** and **is_a()**.

method_exists

(PHP 4 , PHP 5)

method_exists -- Comprueba que el metódo de clase existe

Descripción

bool **method_exists** (object object, string method_name)

Esta función devuelve verdadero (**TRUE**) si el metódo referido por *method_name* ha sido definido en el objecto *object*, en cualquier otro caso devuelve falso (**FALSE**)

XII. Funciones COM y .Net (Windows)

Introducción

COM es un acrónimo para Component Object Model (Modelo de Objetos por Componentes); es una capa orientada a objetos (asi como servicios asociados) que cubre la especificación DCE RPC (un estándar abierto) y define una convención común de llamado que permite que código escrito en cualquier lenguaje pueda llamar e inter-operar con código escrito en cualquier otro lenguaje (provisto que ambos lenguajes hagan uso de COM). No solo es posible escribir el código en cualquier lenguaje, también es cierto que no necesita ser parte del mismo ejecutable; el código puede ser cargado desde un recurso DLL, encontrarse en otro proceso corriendo en la misma máquina, o, mediante DCOM (COM Distribuido), encontrarse en otro proceso en una máquina remota, todo esto sin requerir que su código sepa siquiera en dónde reside el componente.

Existe un sub-conjunto de COM conocido como Automatización OLE que se compone de un grupo de interfaces OLE que permiten los enlaces flexibles con objetos COM, de modo que puedan ser susceptibles a introspección y llamados en tiempo de ejecución sin conocimientos en tiempo de compilación sobre el modo de operación del objeto. La extensión COM de PHP utiliza las interfaces de Automatización OLE para permitirle crear y llamar objetos compatibles desde sus scripts. Técnicamente hablando, ésta debería ser llamada la "Extensión de Automatización OLE para PHP", ya que no todos los objetos COM son compatibles con OLE.

Ahora bien, ¿porqué querría o debería usar COM? COM es una de las formas principales de unir aplicaciones y componentes en la plataforma Windows; mediante el usa de COM usted puede iniciar Microsoft Word, llenar una plantilla de documento y guardar el resultado como un documento Word y enviarlo a un visitante de su sitio web. También puede usar COM para realizar tareas administrativas para su red y para configurar su servidor web (IIS); tales son apenas los usos más comunes; usted puede hacer mucho más con COM.

A partir de PHP 5, esta extensión (y su documentación) fue re-escrita por completo y se ha eliminado gran parte del material confuso e inútil. Adicionalmente, se ofrece soporte para la creación de instancias y ensambles .Net usando la capa de interoperabilidad COM ofrecida por Microsoft.

Por favor lea este artículo para una vista general de los cambios en ésta extensión en PHP 5.

Requirimientos

Las funciones COM se encuentran disponibles únicamente para la versión Windows de PHP.

El soporte para .Net requiere PHP 5 y el entorno de desarrollo .Net.

Instalación

No se necesita ninguna instalación para usar estas funciones, son parte del núcleo de PHP.

La versión para Windows de *PHP* tiene soporte nativo para esta extensión. No se necesita cargar ninguna extensión adicional para usar estas funciones.

Usted es responsable de la instalación del soporte para los varios objetos COM que piensa usar (tales como MS Word); nosotros no incluimos todos éstos con PHP, ni podemos hacerlo.

For Each

A partir de PHP 5, usted puede usar la sentencia la sección de nombre *foreach* en Capítulo 16 de PHP para iterar sobre los contenidos de un IEnumVariant COM/OLE estándar. En términos más simples, esto quiere decir que puede usar foreach en aquellas situaciones en donde podría haber usado *For Each* en código VB/ASP.

Ejemplo 1. For Each en ASP

```
<%
Set objetoDominio = GetObject("WinNT://Domain")
For Each obj in objetoDominio
  Response.Write obj.Name & "<br />"
Next
%>
```

Ejemplo 2. while() ... Next() en PHP 4

```
<?php
$objetoDominio = new COM("WinNT://Domain");
while ($obj = $objetoDominio->Next()) {
   echo $obj->Name . "<br />";
}
?>
```

Ejemplo 3. foreach en PHP 5

```
<?php
$objetoDominio = new COM("WinNT://Domain");
foreach ($objetoDominio as $obj) {
   echo $obj->Name . "<br />";
}
?>
```

Matrices y propiedades COM tipo-matriz

Muchos objetos COM exponen sus propiedades como matrices, o usando un acceso estilo-matriz. En PHP 4, es posible usar la sintaxis de matrices de PHP para leer/escribir tales propiedades, pero sólo es posible manipular una dimensión. Si desea leer una propiedad multi-dimensional, puede crear el acceso en forma de un llamado de función, en donde cada parámetro representa cada parámetro del acceso a la matriz, aunque no hay forma de escribir tal tipo de propiedad.

PHP 5 introduce las siguientes características nuevas para facilitar su vida:

- Acceso a matrices multi-dimensionales, o propiedades COM que requieren múltiples parámetros usando la sintaxis de matrices de PHP. También puede escribir o definir propiedades usando ésta técnica.

- Iterar a través de SafeArrays ("verdaderas" matrices) usando la estructura de control la sección de nombre *foreach* en Capítulo 16. Esto funciona ya que los SafeArrays incluyen información sobre su tamaño. Si una propiedad estilo-matriz implementa IEnumVariant, entonces también puede usar foreach para tales propiedades; eche un vistazo a la sección de nombre *For Each* para más información sobre este tema.

Excepciones (PHP 5)

Esta extensión arroja instancias de la clase *com_exception* siempre que se presente un error potencialmente fatal reportado por COM. Todas las excepciones COM tienen una propiedad *code* bien definida que corresponde con el valor de retorno HRESULT proveniente de las varias operaciones COM. Es posible usar éste código para tomar decisiones programáticas sobre cómo manejar la excepción.

Configuración en tiempo de ejecución

El comportamiento de estas funciones está afectado por los valores definidos en `php.ini`.

Tabla 1. Com configuration options

Name	Default	Changeable
com.allow_dcom	"0"	PHP_INI_SYSTEM
com.autoregister_typelib	"0"	PHP_INI_ALL
com.autoregister_verbose	"0"	PHP_INI_ALL
com.autoregister_casesensitive	"1"	PHP_INI_ALL
com.code_page	""	PHP_INI_ALL
com.typelib_file	""	PHP_INI_SYSTEM

For further details and definitions of the PHP_INI_* constants, see the Apéndice H.

A continuación se presenta una corta explicación de las directivas de configuración.

com.allow_dcom

When this is turned on, PHP will be allowed to operate as a D-COM (Distributed COM) client and will allow the PHP script to instantiate COM objects on a remote server.

com.autoregister_typelib

When this is turned on, PHP will attempt to register constants from the typelibrary of objects that it instantiates, if those objects implement the interfaces required to obtain that information. The case sensitivity of the constants it registers is controlled by the *com.autoregister_casesensitive* configuration directive.

com.autoregister_verbose

When this is turned on, any problems with loading a typelibrary during object instantiation will be reported using the PHP error mechanism. The default is off, which does not emit any indication if there was an error finding or loading the type library.

com.autoregister_casesensitive

When this is turned on (the default), constants found in auto-loaded type libraries will be

registered case sensitively. See **com_load_typelib()** for more details.

com.code_page

It controls the default character set code-page to use when passing strings to and from COM objects. If set to an empty string, PHP will assume that you want **CP_ACP**, which is the default system ANSI code page.

If the text in your scripts is encoded using a different encoding/character set by default, setting this directive will save you from having to pass the code page as a parameter to the COM class constructor. Please note that by using this directive (as with any PHP configuration directive), your PHP script becomes less portable; you should use the COM constructor parameter whenever possible.

Nota: This configuration directive was introduced with PHP 5.

com.typelib_file

When set, this should hold the path to a file that contains a list of typelibraries that should be loaded on startup. Each line of the file will be treated as the type library name and loaded as though you had called **com_load_typelib()**. The constants will be registered persistently, so that the library only needs to be loaded once. If a type library name ends with the string *#cis* or *#case_insensitive*, then the constants from that library will be registered case insensitively.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

CLSCTX_INPROC_SERVER (**integer**)

CLSCTX_INPROC_HANDLER (**integer**)

CLSCTX_LOCAL_SERVER (**integer**)

CLSCTX_REMOTE_SERVER (**integer**)

CLSCTX_SERVER (**integer**)

CLSCTX_ALL (**integer**)

VT_NULL (**integer**)

VT_EMPTY (**integer**)

VT_UI1 (**integer**)

VT_I2 (**integer**)

VT_I4 (**integer**)

VT_R4 (integer)

VT_R8 (integer)

VT_BOOL (integer)

VT_ERROR (integer)

VT_CY (integer)

VT_DATE (integer)

VT_BSTR (integer)

VT_DECIMAL (integer)

VT_UNKNOWN (integer)

VT_DISPATCH (integer)

VT_VARIANT (integer)

VT_I1 (integer)

VT_UI2 (integer)

VT_UI4 (integer)

VT_INT (integer)

VT_UINT (integer)

VT_ARRAY (integer)

VT_BYREF (integer)

CP_ACP (integer)

CP_MACCP (integer)

CP_OEMCP (integer)

CP_UTF7 (integer)

CP_UTF8 (integer)

CP_SYMBOL (integer)

CP_THREAD_ACP (integer)

VARCMP_LT (integer)

VARCMP_EQ (**integer**)

VARCMP_GT (**integer**)

VARCMP_NULL (**integer**)

NORM_IGNORECASE (**integer**)

NORM_IGNORENONSPACE (**integer**)

NORM_IGNORESYMBOLS (**integer**)

NORM_IGNOREWIDTH (**integer**)

NORM_IGNOREKANATYPE (**integer**)

NORM_IGNOREKASHIDA (**integer**)

DISP_E_DIVBYZERO (**integer**)

DISP_E_OVERFLOW (**integer**)

MK_E_UNAVAILABLE (**integer**)

Ver también

Para más información sobre COM, lea la especificación COM o quizás eche un vistazo a la Otra Biblioteca COM Más (YACL por sus siglas en Inglés) de Don Box. Puede encontrar información adicional en nuestro FAQ sobre Capítulo 72. Si está pensando en usar aplicaciones MS Office en el lado del servidor, es buena idea que lea la información encontrada aquí: Consideraciones para la Automatización de Office en el Lado del Servidor.

Tabla de contenidos

com_set -- ???
variant_abs -- Returns the absolute value of a variant
variant_add -- "Adds" two variant values together and returns the result
variant_and -- performs a bitwise AND operation between two variants and returns the result
variant_cast -- Convert a variant into a new variant object of another type
variant_cat -- concatenates two variant values together and returns the result
variant_cmp -- Compares two variants
variant_date_from_timestamp -- Returns a variant date representation of a unix timestamp
variant_date_to_timestamp -- Converts a variant date/time value to unix timestamp
variant_div -- Returns the result from dividing two variants
variant_eqv -- Performs a bitwise ñalence on two variants
variant_fix -- Returns the integer portion ? of a variant
variant_get_type -- Returns the type of a variant object
variant_idiv -- Converts variants to integers and then returns the result from dividing them
variant_imp -- Performs a bitwise implication on two variants
variant_int -- Returns the integer portion of a variant
variant_mod -- Divides two variants and returns only the remainder
variant_mul -- multiplies the values of the two variants and returns the result
variant_neg -- Performs logical negation on a variant
variant_not -- Performs bitwise not negation on a variant
variant_or -- Performs a logical disjunction on two variants
variant_pow -- Returns the result of performing the power function with two variants
variant_round -- Rounds a variant to the specified number of decimal places
variant_set_type -- Convert a variant into another type "in-place"
variant_set -- Assigns a new value for a variant object
variant_sub -- subtracts the value of the right variant from the left variant value and returns the result
variant_xor -- Performs a logical exclusion on two variants

COM

(no version information, might be only in CVS)

COM -- COM class

Sinopsis

```
$obj = new COM("Application.ID")
```

Description

The COM class allows you to instantiate an OLE compatible COM object and call its methods and access its properties.

Methods

com **COM::COM** (string module_name [, mixed server_name [, int codepage [, string typelib]]])

COM class constructor. The parameters have the following meanings:

module_name

Can be a ProgID, Class ID or Moniker that names the component to load.

A ProgID is typically the application or DLL name, followed by a period, followed by the object name. e.g: *Word.Application*.

A Class ID is the UUID that uniquely identifies a given class.

A Moniker is a special form of naming, similar in concept to a URL scheme, that identifies a resource and specifies how it should be loaded. As an example, you could load up Word and get an object representing a word document by specifying the full path to the word document as the module name, or you can use *LDAP:* as a moniker to use the ADSI interface to LDAP.

server_name

The name of the DCOM server on which the component should be loaded and run. If **NULL**, the object is run using the default for the application. The default is typically to run it on the local machine, although the administrator might have configured the application to launch on a different machine.

If you specify a non-**NULL** value for server, PHP will refuse to load the object unless the *com.allow_dcom* configuration option is set to **TRUE**.

If *server_name* is an array, it should contain the following elements (case sensitive!). Note that they are all optional (although you need to specify both Username and Password together); if you omit the Server setting, the default server will be used (as mentioned above), and the instantiation of the object will not be affected by the *com.allow_dcom* directive.

Tabla 1. DCOM server name

***server_name* key**	**type**	**description**
Server	string	The name of the server.
Username	string	The username to connect as.
Password	string	The password for *Username*.
Flags	integer	One or more of the following constants, logically OR'd together: **CLSCTX_INPROC_SERVER**, **CLSCTX_INPROC_HANDLER**, **CLSCTX_LOCAL_SERVER**, **CLSCTX_REMOTE_SERVER**, **CLSCTX_SERVER** and **CLSCTX_ALL**. The default value if not specified here is **CLSCTX_SERVER** if you also omit *Server*, or **CLSCTX_REMOTE_SERVER** if you do specify a server. You should consult the Microsoft documentation for CoCreateInstance for more information on the meaning of these constants; you will typically never have to use them.

codepage

Specifies the codepage that is used to convert strings to unicode-strings and vice versa. The conversion is applied whenever a PHP string is passed as a parameter or returned from a method of this COM object. The code page is sticky in PHP 5, which means that it will propagate to objects and variants returned from the object.

Possible values are **CP_ACP** (use system default ANSI code page - the default if this parameter is omitted), **CP_MACCP**, **CP_OEMCP**, **CP_SYMBOL**, **CP_THREAD_ACP** (use

codepage/locale set for the current executing thread), **CP_UTF7** and **CP_UTF8**. You may also use the number for a given codepage; consult the Microsoft documentation for more details on codepages and their numeric values.

Overloaded Methods

The returned object is an overloaded object, which means that PHP does not see any fixed methods as it does with regular classes; instead, any property or method accesses are passed through to COM.

Starting with PHP 5, PHP will automatically detect methods that accept parameters by reference, and will automatically convert regular PHP variables to a form that can be passed by reference. This means that you can call the method very naturally; you needn't go to any extra effort in your code.

In PHP 4, to pass parameters by reference you need to create an instance of the VARIANT class to wrap the byref parameters.

Pseudo Methods

In PHP versions prior to 5, a number of not very pleasant hacks meant that the following method names were not passed through to COM and were handled directly by PHP. PHP 5 eliminates these things; read the details below to determine how to fix your scripts. These magic method names are case insensitive.

void **COM::AddRef** (void)

Artificially adds a reference count to the COM object.

Aviso
You should never need to use this method. It exists as a logical complement to the Release() method below.

void **COM::Release** (void)

Artificially removes a reference count from the COM object.

Aviso
You should never need to use this method. Its existence in PHP is a bug designed to work around a bug that keeps COM objects running longer than they should.

Pseudo Methods for Iterating

These pseudo methods are only available if **com_isenum()** returns **TRUE**, in which case, they hide any methods with the same names that might otherwise be provided by the COM object. These methods have all been eliminated in PHP 5, and you should use la sección de nombre *For Each* en Referencia XII, *Funciones COM y .Net (Windows)* instead.

variant **COM::All** (void)

Returns a variant representing a SafeArray that has 10 elements; each element will be an empty/null variant. This function was supposed to return an array containing all the elements from the iterator, but was never completed. Do not use.

variant **COM::Next** (void)

Returns a variant representing the next element available from the iterator, or **FALSE** when there are no more elements.

variant **COM::Prev** (void)

Returns a variant representing the previous element available from the iterator, or **FALSE** when there are no more elements.

void **COM::Reset** (void)

Rewinds the iterator back to the start.

COM examples

Ejemplo 1. COM example (1)

```
<?php
// starting word
$word = new COM("word.application") or die("Unable to instantiate Word");
echo "Loaded Word, version {$word->Version}\n";

//bring it to front
$word->Visible = 1;

//open an empty document
$word->Documents->Add();

//do some weird stuff
$word->Selection->TypeText("This is a test...");
$word->Documents[1]->SaveAs("Useless test.doc");

//closing word
$word->Quit();

//free the object
$word = null;
?>
```

Ejemplo 2. COM example (2)

```
<?php

$conn = new COM("ADODB.Connection") or die("Cannot start ADO");
$conn->Open("Provider=SQLOLEDB; Data Source=localhost;
Initial Catalog=database; User ID=user; Password=password");

$rs = $conn->Execute("SELECT * FROM sometable");    // Recordset

$num_columns = $rs->Fields->Count();
echo $num_columns . "\n";

for ($i=0; $i < $num_columns; $i++) {
    $fld[$i] = $rs->Fields($i);
}

$rowcount = 0;
while (!$rs->EOF) {
    for ($i=0; $i < $num_columns; $i++) {
        echo $fld[$i]->value . "\t";
    }
    echo "\n";
    $rowcount++;            // increments rowcount
    $rs->MoveNext();
}

$rs->Close();
$conn->Close();

$rs = null;
$conn = null;

?>
```

DOTNET

(no version information, might be only in CVS)

DOTNET -- DOTNET class

Sinopsis

```
$obj = new DOTNET("assembly", "classname")
```

Description

The DOTNET class allows you to instantiate a class from a .Net assembly and call its methods and access its properties.

Methods

string **DOTNET::DOTNET** (string assembly_name, string class_name [, int codepage])

DOTNET class constructor. *assembly_name* specifies which assembly should be loaded, and *class_name* specifices which class in that assembly to instantiate. You may optionally specify a *codepage* to use for unicode string transformations; see the COM class for more details on code pages.

The returned object is an overloaded object, which means that PHP does not see any fixed methods as it does with regular classes; instead, any property or method accesses are passed through to COM

and from there to DOTNET. In other words, the .Net object is mapped through the COM interoperability layer provided by the .Net runtime.

Once you have created a DOTNET object, PHP treats it identically to any other COM object; all the same rules apply.

Ejemplo 1. DOTNET example

```
<?php
  $stack = new DOTNET("mscorlib", "System.Collections.Stack");
  $stack->Push(".Net");
  $stack->Push("Hello ");
  echo $stack->Pop() . $stack->Pop();
?>
```

> **Nota:** You need to install the .Net runtime on your web server to take advantage of this feature.

VARIANT

(no version information, might be only in CVS)

VARIANT -- VARIANT class

Sinopsis

```
$vVar = new VARIANT($var)
```

Description

The VARIANT is COM's ñalent of the PHP zval; it is a structure that can contain a value with a range of different possible types. The VARIANT class provided by the COM extension allows you to have more control over the way that PHP passes values to and from COM.

Methods

object **VARIANT::VARIANT** ([mixed value [, int type [, int codepage]]])

VARIANT class constructor. Parameters:

value

initial value. if omitted, or set to **NULL** an VT_EMPTY object is created.

type

specifies the content type of the VARIANT object. Possible values are one of the **VT_XXX** la sección de nombre *Constantes predefinidas* en Referencia XII, *Funciones COM y .Net (Windows)*.

In PHP versions prior to PHP 5, you could force PHP to pass a variant object by reference by OR'ing **VT_BYREF** with the *type*. In PHP 5, this hack is not supported; instead, PHP 5 can detect parameters passed by reference automatically; they do not even need to be passed as VARIANT objects.

Consult the MSDN library for additional information on the VARIANT type.

codepage

specifies the codepage that is used to convert strings to unicode. See the parameter of the same name in the COM class for more information.

PHP versions prior to PHP 5 define a number of (undocumented) virtual properties for instances of the VARIANT class; these properties have all been removed in PHP 5 in favour of its more natural syntax; these differences are best highlighted by example:

Ejemplo 1. Variant example, PHP 4.x style

```
<?php
$v = new VARIANT(42);
print "The type is " . $v->type . "<br/>";
print "The value is " . $v->value . "<br/>";
?>
```

Ejemplo 2. Variant example, PHP 5 style

```
<?php
$v = new VARIANT(42);
print "The type is " . variant_get_type($v) . "<br/>";
print "The value is " . $v . "<br/>";
?>
```

The reason for the change is that, internally, the COM extension sees VARIANT, COM and DOTNET classes as the same thing, and the design philosophy for these classes is that all property and member accesses are passed through to COM with no interference. The new syntax is more natural and less effort, and most of the removed virtual properties didn't make any sense in a PHP context in any case.

Nota: PHP 5 takes a much simpler approach to handling VARIANTs; when returning a value or fetching a variant property, the variant is converted to a PHP value only when there is a direct mapping between the types that would not result in a loss of information. In all other cases, the result is returned as an instance of the VARIANT class. You can force PHP to convert or evaluate the variant as a PHP native type by using a casting operator explicitly, or implicitly casting to a string by **print()**ing it. You may use the wide range of variant functions to perform arithmetic operations on variants without forcing a conversion or risking a loss of data.

See also **variant_get_type()**.

com_addref

(PHP 4 >= 4.1.0)

com_addref -- Increases the components reference counter [deprecated]

Description

void **com_addref** (void)

Increases the components reference counter.

Aviso

You should never need to use this function.

Nota: This function has gone away in PHP 5.

com_create_guid

(PHP 5)

com_create_guid -- Generate a globally unique identifier (GUID)

Description

string **com_create_guid** (void)

Generates a Globally Unique Identifier (GUID) and returns it as a string. A GUID is generated in the same way as DCE UUID's, except that the Microsoft convention is to enclose a GUID in curly braces.

See also **uuid_create()** in the PECL uuid extension.

com_event_sink

(PHP 4 >= 4.2.3, PHP 5)

com_event_sink -- Connect events from a COM object to a PHP object

Description

bool **com_event_sink** (variant comobject, object sinkobject [, mixed sinkinterface])

Instructs COM to sink events generated by *comobject* into the PHP object *sinkobject*. PHP will attempt to use the default dispinterface type specified by the typelibrary associated with *comobject*, but you may override this choice by setting *sinkinterface* to the name of the dispinterface that you want to use.

sinkobject should be an instance of a class with methods named after those of the desired dispinterface; you may use **com_print_typeinfo()** to help generate a template class for this purpose.

Be careful how you use this feature; if you are doing something similar to the example below, then it doesn't really make sense to run it in a web server context.

Ejemplo 1. COM event sink example

```
<?php
class IEEventSinker {
  var $terminated = false;

  function ProgressChange($progress, $progressmax) {
    echo "Download progress: $progress / $progressmax\n";
  }

  function DocumentComplete(&$dom, $url) {
    echo "Document $url complete\n";
  }

  function OnQuit() {
    echo "Quit!\n";
    $this->terminated = true;
  }
}
$ie = new COM("InternetExplorer.Application");
// note that you don't need the & for PHP 5!
$sink =& new IEEventSinker();
com_event_sink($ie, $sink, "DWebBrowserEvents2");
$ie->Visible = true;
$ie->Navigate("http://www.php.net");
while(!$sink->terminated) {
  com_message_pump(4000);
}
$ie = null;
?>
```

See also **com_print_typeinfo()**, **com_message_pump()**.

com_get_active_object

(no version information, might be only in CVS)

com_get_active_object -- Returns a handle to an already running instance of a COM object

Description

variant **com_get_active_object** (string progid [, int code_page])

com_get_active_object() is similar to creating a new instance of a COM object, except that it will only return an object to your script if the object is already running. OLE applications use something known as the Running Object Table to allow well-known applications to be launched only once; this function exposes the COM library function GetActiveObject() to get a handle on a running instance.

progid must be either the ProgID or CLSID for the object that you want to access (for example *Word.Application*). *code_page* acts in precisely the same way that it does for the COM class.

If the requested object is running, it will be returned to your script just like any other COM object. Otherwise a *com_exception* will be raised. There are a variety of reasons why this function might fail, the most common being that the object is not already running. In that situation, the exception error code will be **MK_E_UNAVAILABLE**; you can use the *getCode* method of the exception object to check the exception code.

Aviso

Using **com_get_active_object()** in a web server context is not always a smart idea. Most COM/OLE applications are not designed to handle more than one client concurrently, even (or especially!) Microsoft Office. You should read Considerations for Server-Side Automation of Office for more information on the general issues involved.

com_get

(PHP 3>= 3.0.3, PHP 4 >= 4.0.5)

com_get -- ???

Descripción

mixed **com_get** (resource object, string property)

com_invoke

(PHP 3>= 3.0.3)

com_invoke -- ???

Descripción

mixed **com_invoke** (resource object, string function_name [, mixed function parameters, ...])

com_isenum

(PHP 4 >= 4.1.0)

com_isenum -- Indicates if a COM object has an IEnumVariant interface for iteration [deprecated]

Description

bool **com_isenum** (variant com_module)

Checks to see if a COM object can be enumerated using the *Next()* method hack. Returns **TRUE** if it can, **FALSE** if it cannot. See COM class for more details on these methods.

> **Nota:** This function does not exist in PHP 5; use the more natural la sección de nombre *foreach* en Capítulo 16 statement to iterate over the contents of COM objects. See la sección de nombre *For Each* en Referencia XII, *Funciones COM y .Net (Windows)* for more details.

com_load_typelib

(PHP 4 >= 4.1.0, PHP 5)

com_load_typelib -- Loads a Typelib

Description

bool **com_load_typelib** (string typelib_name [, bool case_insensitive])

Loads a type-library and registers its constants in the engine, as though they were defined using **define()**. The *case_insensitive* behaves in the same way as the parameter with the same name in the **define()** function.

typelib_name can be one of the following:

- The filename of a *.tlb* file or the executable module that contains the type library.
- The type library GUID, followed by its version number, for example *{00000200-0000-0010-8000-00AA006D2EA4},2,0*.
- The type library name, e.g. *Microsoft OLE DB ActiveX Data Objects 1.0 Library*.

PHP will attempt to resolve the type library in this order, as the process gets more and more expensive as you progress down the list; searching for the type library by name is handled by physically enumerating the registry until we find a match.

Note that it is much more efficient to use the *com.typelib_file* configuration setting to pre-load and register the constants, although not so flexible.

If you have turned on *com.autoregister_typelib*, then PHP will attempt to automatically register the constants associated with a COM object when you instantiate it. This depends on the interfaces provided by the COM object itself, and may not always be possible.

com_load

(PHP 3>= 3.0.3)

com_load -- ???

Descripción

string **com_load** (string module name [, string server name])

com_message_pump

(PHP 4 >= 4.2.3, PHP 5)

com_message_pump -- Process COM messages, sleeping for up to timeoutms milliseconds

Description

bool **com_message_pump** ([int timeoutms])

This function will sleep for up to *timeoutms* milliseconds, or until a message arrives in the queue. If a message or messages arrives before the timeout, they will be dispatched, and the function will return **TRUE**. If the timeout occurs and no messages were processed, the return value will be

FALSE. If you do not specify a value for *timeoutms*, then 0 will be assumed. A 0 value means that no waiting will be performed; if there are messages pending they will be dispatched as before; if there are no messages pending, the function will return **FALSE** immediately without sleeping.

The purpose of this function is to route COM calls between apartments and handle various synchronization issues. This allows your script to wait efficiently for events to be triggered, while still handling other events or running other code in the background. You should use it in a loop, as demonstrated by the example in the **com_event_sink()** function, until you are finished using event bound COM objects.

com_print_typeinfo

(PHP 4 >= 4.2.3, PHP 5)

com_print_typeinfo -- Print out a PHP class definition for a dispatchable interface

Description

bool **com_print_typeinfo** (object comobject [, string dispinterface [, bool wantsink]])

The purpose of this function is to help generate a skeleton class for use as an event sink. You may also use it to generate a dump of any COM object, provided that it supports enough of the introspection interfaces, and that you know the name of the interface you want to display.

comobject should be either an instance of a COM object, or be the name of a typelibrary (which will be resolved according to the rules set out in **com_load_typelib()**). *dispinterface* is the name of an IDispatch descendant interface that you want to display. If *wantsink* is **TRUE**, the corresponding sink interface will be displayed instead.

See also **com_event_sink()**, **com_load_typelib()**.

com_propget

(PHP 3>= 3.0.3, PHP 4 >= 4.0.5)

com_propget -- ???

Descripción

mixed **com_propget** (resource object, string property)

com_propput

(PHP 3>= 3.0.3, PHP 4 >= 4.0.5)

com_propput -- ???

Descripción

void **com_propput** (resource object, string property, mixed value)

com_propset

(PHP 3>= 3.0.3, PHP 4 >= 4.0.5)

com_propset -- ???

Descripción

void **com_propset** (resource object, string property, mixed value)

Esta función es un alias para **com_propput()**.

com_release

(PHP 4 >= 4.1.0)

com_release -- Decreases the components reference counter [deprecated]

Description

void **com_release** (void)

Decreases the components reference counter.

Aviso
You should never need to use this function.

Nota: This function has gone away in PHP 5.

com_set

(PHP 3>= 3.0.3, PHP 4 >= 4.0.5)

com_set -- ???

Descripción

void **com_set** (resource object, string property, mixed value)

Esta función es un alias para **com_set()**.

variant_abs

(PHP 5)

variant_abs -- Returns the absolute value of a variant

Description

mixed **variant_abs** (mixed val)

Returns the absolute value of *val*.

See also **abs()**.

> **Nota:** As with all the variant arithmetic functions, the parameters for this function can be either a PHP native type (integer, string, floating point, boolean or **NULL**), or an instance of a COM, VARIANT or DOTNET class. PHP native types will be converted to variants using the same rules as found in the constructor for the VARIANT class. COM and DOTNET objects will have the value of their default property taken and used as the variant value.
>
> The variant arithmetic functions are wrappers around the similarly named functions in the COM library; for more information on these functions, consult the MSDN library. The PHP functions are named slightly differently; for example variant_add() in PHP corresponds to VarAdd() in the MSDN documentation.

variant_add

(PHP 5)

variant_add -- "Adds" two variant values together and returns the result

Description

mixed **variant_add** (mixed left, mixed right)

Adds *left* to *right* using the following rules (taken from the MSDN library), which correspond to those of Visual Basic:

Tabla 1. Variant Addition Rules

If	Then
Both expressions are of the string type	Concatenation
One expression is a string type and the other a character	Addition
One expression is numeric and the other is a string	Addition
Both expressions are numeric	Addition
Either expression is NULL	NULL is returned
Both expressions are empty	Integer subtype is returned

See also **variant_sub()**.

> **Nota:** As with all the variant arithmetic functions, the parameters for this function can

be either a PHP native type (integer, string, floating point, boolean or **NULL**), or an instance of a COM, VARIANT or DOTNET class. PHP native types will be converted to variants using the same rules as found in the constructor for the VARIANT class. COM and DOTNET objects will have the value of their default property taken and used as the variant value.

The variant arithmetic functions are wrappers around the similarly named functions in the COM library; for more information on these functions, consult the MSDN library. The PHP functions are named slightly differently; for example variant_add() in PHP corresponds to VarAdd() in the MSDN documentation.

variant_and

(PHP 5)

variant_and -- performs a bitwise AND operation between two variants and returns the result

Description

mixed **variant_and** (mixed left, mixed right)

Performs a bitwise AND operation, according to the following truth table; note that this is slightly different from a regular AND operation.

Tabla 1. Variant AND Rules

If *left* is	If *right* is	then the result is
TRUE	TRUE	TRUE
TRUE	FALSE	FALSE
TRUE	NULL	NULL
FALSE	TRUE	FALSE
FALSE	FALSE	FALSE
FALSE	NULL	FALSE
NULL	TRUE	NULL
NULL	FALSE	FALSE
NULL	NULL	NULL

See also **variant_or()**.

Nota: As with all the variant arithmetic functions, the parameters for this function can be either a PHP native type (integer, string, floating point, boolean or **NULL**), or an instance of a COM, VARIANT or DOTNET class. PHP native types will be converted to variants using the same rules as found in the constructor for the VARIANT class. COM and DOTNET objects will have the value of their default property taken and used as the variant value.

The variant arithmetic functions are wrappers around the similarly named functions in the COM library; for more information on these functions, consult the MSDN library. The PHP functions are named slightly differently; for example variant_add() in PHP

corresponds to VarAdd() in the MSDN documentation.

variant_cast

(PHP 5)

variant_cast -- Convert a variant into a new variant object of another type

Description

variant **variant_cast** (variant variant, int type)

This function makes a copy of *variant* and then performs a variant cast operation to force the copy to have the type given by *type*. *type* should be one of the **VT_XXX** constants.

This function wraps VariantChangeType() in the COM library; consult MSDN for more information.

See also **variant_set_type()**.

variant_cat

(PHP 5)

variant_cat -- concatenates two variant values together and returns the result

Description

mixed **variant_cat** (mixed left, mixed right)

Concatenates *left* with *right* and returns the result.

See also la sección de nombre *Operadores de Cadena* en Capítulo 15 for the string concatenation operator; this function is notionally ñalent to *$left . $right*.

> **Nota:** As with all the variant arithmetic functions, the parameters for this function can be either a PHP native type (integer, string, floating point, boolean or **NULL**), or an instance of a COM, VARIANT or DOTNET class. PHP native types will be converted to variants using the same rules as found in the constructor for the VARIANT class. COM and DOTNET objects will have the value of their default property taken and used as the variant value.
>
> The variant arithmetic functions are wrappers around the similarly named functions in the COM library; for more information on these functions, consult the MSDN library. The PHP functions are named slightly differently; for example variant_add() in PHP corresponds to VarAdd() in the MSDN documentation.

variant_cmp

(PHP 5)

variant_cmp -- Compares two variants

Description

int **variant_cmp** (mixed left, mixed right [, int lcid [, int flags]])

Compares *left* with *right* and returns one of the following values:

Tabla 1. Variant Comparision Results

value	meaning
VARCMP_LT	*left* is less than *right*
VARCMP_EQ	*left* is equal to *right*
VARCMP_GT	*left* is greater than *right*
VARCMP_NULL	Either *left*, *right* or both are **NULL**

This function will only compare scalar values, not arrays or variant records.

lcid is a valid Locale Identifier to use when comparing strings (this affects string collation). *flags* can be one or more of the following values OR'd together, and affects string comparisons:

Tabla 2. Variant Comparision Flags

value	meaning
NORM_IGNORECASE	Compare case insensitively
NORM_IGNORENONSPACE	Ignore nonspacing characters
NORM_IGNORESYMBOLS	Ignore symbols
NORM_IGNOREWIDTH	Ignore string width
NORM_IGNOREKANATYPE	Ignore Kana type
NORM_IGNOREKASHIDA	Ignore Arabic kashida characters

Nota: As with all the variant arithmetic functions, the parameters for this function can be either a PHP native type (integer, string, floating point, boolean or **NULL**), or an instance of a COM, VARIANT or DOTNET class. PHP native types will be converted to variants using the same rules as found in the constructor for the VARIANT class. COM and DOTNET objects will have the value of their default property taken and used as the variant value.

The variant arithmetic functions are wrappers around the similarly named functions in the COM library; for more information on these functions, consult the MSDN library. The PHP functions are named slightly differently; for example variant_add() in PHP corresponds to VarAdd() in the MSDN documentation.

variant_date_from_timestamp

(PHP 5)

variant_date_from_timestamp -- Returns a variant date representation of a unix timestamp

Description

variant **variant_date_from_timestamp** (int timestamp)

Converts *timestamp* from a unix timestamp value into a variant of type **VT_DATE**. This allows easier interopability between the unix-ish parts of PHP and COM.

See also **variant_date_to_timestamp()** for the inverse of this operation, **mktime()**, **time()**.

variant_date_to_timestamp

(PHP 5)

variant_date_to_timestamp -- Converts a variant date/time value to unix timestamp

Description

int **variant_date_to_timestamp** (variant variant)

Converts *variant* from a **VT_DATE** (or similar) value into a unix timestamp. This allows easier interopability between the unix-ish parts of PHP and COM.

See also **variant_date_from_timestamp()** for the inverse of this operation, **date()**, **strftime()**.

variant_div

(PHP 5)

variant_div -- Returns the result from dividing two variants

Description

mixed **variant_div** (mixed left, mixed right)

Divides *left* by *right* and returns the result, subject to the following rules:

Tabla 1. Variant Division Rules

If	Then
Both expressions are of the string, date, character, boolean type	Double is returned
One expression is a string type and the other a character	Division and a double is returned
One expression is numeric and the other is a string	Division and a double is returned.
Both expressions are numeric	Division and a double is returned
Either expression is NULL	NULL is returned

If	Then
right is empty and *left* is anything but empty	A com_exception with code **DISP_E_DIVBYZERO** is thrown
left is empty and *right* is anything but empty.	0 as type double is returned
Both expressions are empty	A com_exception with code **DISP_E_OVERFLOW** is thrown

Nota: As with all the variant arithmetic functions, the parameters for this function can be either a PHP native type (integer, string, floating point, boolean or **NULL**), or an instance of a COM, VARIANT or DOTNET class. PHP native types will be converted to variants using the same rules as found in the constructor for the VARIANT class. COM and DOTNET objects will have the value of their default property taken and used as the variant value.

The variant arithmetic functions are wrappers around the similarly named functions in the COM library; for more information on these functions, consult the MSDN library. The PHP functions are named slightly differently; for example variant_add() in PHP corresponds to VarAdd() in the MSDN documentation.

variant_eqv

(PHP 5)

variant_eqv -- Performs a bitwise ñalence on two variants

Description

mixed **variant_eqv** (mixed left, mixed right)

If each bit in *left* is equal to the corresponding bit in *right* then **TRUE** is returned, otherwise **FALSE** is returned.

Nota: As with all the variant arithmetic functions, the parameters for this function can be either a PHP native type (integer, string, floating point, boolean or **NULL**), or an instance of a COM, VARIANT or DOTNET class. PHP native types will be converted to variants using the same rules as found in the constructor for the VARIANT class. COM and DOTNET objects will have the value of their default property taken and used as the variant value.

The variant arithmetic functions are wrappers around the similarly named functions in the COM library; for more information on these functions, consult the MSDN library. The PHP functions are named slightly differently; for example variant_add() in PHP corresponds to VarAdd() in the MSDN documentation.

variant_fix

(PHP 5)

variant_fix -- Returns the integer portion ? of a variant

Description

mixed **variant_fix** (mixed variant)

If *variant* is negative, then the first negative integer greater than or equal to the variant is returned, otherwise returns the integer portion of the value of *variant*.

See also **variant_int()**, **variant_round()**, **floor()**, **ceil()**, **round()**.

Aviso
This documentation is based on the MSDN documentation; it appears that this function is either the same as **variant_int()**, or that there is an error in the MSDN documentation.

> **Nota:** As with all the variant arithmetic functions, the parameters for this function can be either a PHP native type (integer, string, floating point, boolean or **NULL**), or an instance of a COM, VARIANT or DOTNET class. PHP native types will be converted to variants using the same rules as found in the constructor for the VARIANT class. COM and DOTNET objects will have the value of their default property taken and used as the variant value.
>
> The variant arithmetic functions are wrappers around the similarly named functions in the COM library; for more information on these functions, consult the MSDN library. The PHP functions are named slightly differently; for example variant_add() in PHP corresponds to VarAdd() in the MSDN documentation.

variant_get_type

(PHP 5)

variant_get_type -- Returns the type of a variant object

Description

int **variant_get_type** (variant variant)

This function returns an integer value that indicates the type of *variant*, which can be an instance of COM, DOTNET or VARIANT classes. The return value can be compared to one of the **VT_XXX** constants.

The return value for COM and DOTNET objects will usually be **VT_DISPATCH**; the only reason this function works for those classes is because COM and DOTNET are descendants of VARIANT.

In PHP versions prior to 5, you could obtain this information from instances of the VARIANT class ONLY, by reading a fake *type* property. See the VARIANT class for more information on this.

variant_idiv

(PHP 5)

variant_idiv -- Converts variants to integers and then returns the result from dividing them

Description

mixed **variant_idiv** (mixed left, mixed right)

Converts *left* and *right* to integer values, and then performs integer division according the following rules:

Tabla 1. Variant Integer Division Rules

If	Then
Both expressions are of the string, date, character, boolean type	Division and integer is returned
One expression is a string type and the other a character	Division
One expression is numeric and the other is a string	Division
Both expressions are numeric	Division
Either expression is NULL	NULL is returned
Both expressions are empty	A com_exception with code **DISP_E_DIVBYZERO** is thrown

See also **variant_div()**.

> **Nota:** As with all the variant arithmetic functions, the parameters for this function can be either a PHP native type (integer, string, floating point, boolean or **NULL**), or an instance of a COM, VARIANT or DOTNET class. PHP native types will be converted to variants using the same rules as found in the constructor for the VARIANT class. COM and DOTNET objects will have the value of their default property taken and used as the variant value.
>
> The variant arithmetic functions are wrappers around the similarly named functions in the COM library; for more information on these functions, consult the MSDN library. The PHP functions are named slightly differently; for example variant_add() in PHP corresponds to VarAdd() in the MSDN documentation.

variant_imp

(PHP 5)

variant_imp -- Performs a bitwise implication on two variants

Description

mixed **variant_imp** (mixed left, mixed right)

Performs a bitwise implication operation, according to the following truth table:

Tabla 1. Variant Implication Table

If *left* is	If *right* is	then the result is
TRUE	**TRUE**	**TRUE**
TRUE	**FALSE**	**TRUE**
TRUE	**NULL**	**TRUE**
FALSE	**TRUE**	**TRUE**
FALSE	**FALSE**	**TRUE**
FALSE	**NULL**	**TRUE**
NULL	**TRUE**	**TRUE**
NULL	**FALSE**	**NULL**
NULL	**NULL**	**NULL**

Nota: As with all the variant arithmetic functions, the parameters for this function can be either a PHP native type (integer, string, floating point, boolean or **NULL**), or an instance of a COM, VARIANT or DOTNET class. PHP native types will be converted to variants using the same rules as found in the constructor for the VARIANT class. COM and DOTNET objects will have the value of their default property taken and used as the variant value.

The variant arithmetic functions are wrappers around the similarly named functions in the COM library; for more information on these functions, consult the MSDN library. The PHP functions are named slightly differently; for example variant_add() in PHP corresponds to VarAdd() in the MSDN documentation.

variant_int

(PHP 5)

variant_int -- Returns the integer portion of a variant

Description

mixed **variant_int** (mixed variant)

If *variant* is negative, then the first negative integer greater than or equal to the variant is returned, otherwise returns the integer portion of the value of *variant*.

See also **variant_fix()**, **variant_round()**, **floor()**, **ceil()**, **round()**.

Nota: As with all the variant arithmetic functions, the parameters for this function can be either a PHP native type (integer, string, floating point, boolean or **NULL**), or an instance of a COM, VARIANT or DOTNET class. PHP native types will be converted to variants using the same rules as found in the constructor for the VARIANT class. COM and DOTNET objects will have the value of their default property taken and used as the variant value.

The variant arithmetic functions are wrappers around the similarly named functions in the COM library; for more information on these functions, consult the MSDN library. The PHP functions are named slightly differently; for example variant_add() in PHP corresponds to VarAdd() in the MSDN documentation.

variant_mod

(PHP 5)

variant_mod -- Divides two variants and returns only the remainder

Description

mixed **variant_mod** (mixed left, mixed right)

Divides *left* by *right* and returns the remainder.

See also **variant_div()**, **variant_idiv()**.

> **Nota:** As with all the variant arithmetic functions, the parameters for this function can be either a PHP native type (integer, string, floating point, boolean or **NULL**), or an instance of a COM, VARIANT or DOTNET class. PHP native types will be converted to variants using the same rules as found in the constructor for the VARIANT class. COM and DOTNET objects will have the value of their default property taken and used as the variant value.
>
> The variant arithmetic functions are wrappers around the similarly named functions in the COM library; for more information on these functions, consult the MSDN library. The PHP functions are named slightly differently; for example variant_add() in PHP corresponds to VarAdd() in the MSDN documentation.

variant_mul

(PHP 5)

variant_mul -- multiplies the values of the two variants and returns the result

Description

mixed **variant_mul** (mixed left, mixed right)

Multiplies *left* by *right* and returns the result, subject to the following rules:

Tabla 1. Variant Multiplication Rules

If	Then
Both expressions are of the string, date, character, boolean type	Multiplication
One expression is a string type and the other a character	Multiplication
One expression is numeric and the other is a string	Multiplication
Both expressions are numeric	Multiplication
Either expression is NULL	NULL is returned

If	Then
Both expressions are empty	Empty string is returned

Boolean values are converted to -1 for **FALSE** and 0 for **TRUE**.

See also **variant_div()**, **variant_idiv()**.

> **Nota:** As with all the variant arithmetic functions, the parameters for this function can be either a PHP native type (integer, string, floating point, boolean or **NULL**), or an instance of a COM, VARIANT or DOTNET class. PHP native types will be converted to variants using the same rules as found in the constructor for the VARIANT class. COM and DOTNET objects will have the value of their default property taken and used as the variant value.
>
> The variant arithmetic functions are wrappers around the similarly named functions in the COM library; for more information on these functions, consult the MSDN library. The PHP functions are named slightly differently; for example variant_add() in PHP corresponds to VarAdd() in the MSDN documentation.

variant_neg

(PHP 5)

variant_neg -- Performs logical negation on a variant

Description

mixed **variant_neg** (mixed variant)

Performs logical negation of *variant* and returns the result.

> **Nota:** As with all the variant arithmetic functions, the parameters for this function can be either a PHP native type (integer, string, floating point, boolean or **NULL**), or an instance of a COM, VARIANT or DOTNET class. PHP native types will be converted to variants using the same rules as found in the constructor for the VARIANT class. COM and DOTNET objects will have the value of their default property taken and used as the variant value.
>
> The variant arithmetic functions are wrappers around the similarly named functions in the COM library; for more information on these functions, consult the MSDN library. The PHP functions are named slightly differently; for example variant_add() in PHP corresponds to VarAdd() in the MSDN documentation.

variant_not

(PHP 5)

variant_not -- Performs bitwise not negation on a variant

Description

mixed **variant_not** (mixed variant)

Performs bitwise not negation on *variant* and returns the result. If *variant* is **NULL**, the result will also be **NULL**.

> **Nota:** As with all the variant arithmetic functions, the parameters for this function can be either a PHP native type (integer, string, floating point, boolean or **NULL**), or an instance of a COM, VARIANT or DOTNET class. PHP native types will be converted to variants using the same rules as found in the constructor for the VARIANT class. COM and DOTNET objects will have the value of their default property taken and used as the variant value.
>
> The variant arithmetic functions are wrappers around the similarly named functions in the COM library; for more information on these functions, consult the MSDN library. The PHP functions are named slightly differently; for example variant_add() in PHP corresponds to VarAdd() in the MSDN documentation.

variant_or

(PHP 5)

variant_or -- Performs a logical disjunction on two variants

Description

mixed **variant_or** (mixed left, mixed right)

Performs a bitwise OR operation, according to the following truth table; note that this is slightly different from a regular OR operation.

Tabla 1. Variant OR Rules

If *left* is	If *right* is	then the result is
TRUE	**TRUE**	**TRUE**
TRUE	**FALSE**	**TRUE**
TRUE	**NULL**	**TRUE**
FALSE	**TRUE**	**TRUE**
FALSE	**FALSE**	**FALSE**
FALSE	**NULL**	**NULL**
NULL	**TRUE**	**TRUE**
NULL	**FALSE**	**NULL**
NULL	**NULL**	**NULL**

See also **variant_and()**, **variant_xor()**.

> **Nota:** As with all the variant arithmetic functions, the parameters for this function can

be either a PHP native type (integer, string, floating point, boolean or **NULL**), or an instance of a COM, VARIANT or DOTNET class. PHP native types will be converted to variants using the same rules as found in the constructor for the VARIANT class. COM and DOTNET objects will have the value of their default property taken and used as the variant value.

The variant arithmetic functions are wrappers around the similarly named functions in the COM library; for more information on these functions, consult the MSDN library. The PHP functions are named slightly differently; for example variant_add() in PHP corresponds to VarAdd() in the MSDN documentation.

variant_pow

(PHP 5)

variant_pow -- Returns the result of performing the power function with two variants

Description

mixed **variant_pow** (mixed left, mixed right)

Returns the result of *left* to the power of *right*.

See also **pow()**.

Nota: As with all the variant arithmetic functions, the parameters for this function can be either a PHP native type (integer, string, floating point, boolean or **NULL**), or an instance of a COM, VARIANT or DOTNET class. PHP native types will be converted to variants using the same rules as found in the constructor for the VARIANT class. COM and DOTNET objects will have the value of their default property taken and used as the variant value.

The variant arithmetic functions are wrappers around the similarly named functions in the COM library; for more information on these functions, consult the MSDN library. The PHP functions are named slightly differently; for example variant_add() in PHP corresponds to VarAdd() in the MSDN documentation.

variant_round

(PHP 5)

variant_round -- Rounds a variant to the specified number of decimal places

Description

mixed **variant_round** (mixed variant, int decimals)

Returns the value of *variant* rounded to *decimals* decimal places.

See also **round()**.

Nota: As with all the variant arithmetic functions, the parameters for this function can be either a PHP native type (integer, string, floating point, boolean or **NULL**), or an instance of a COM, VARIANT or DOTNET class. PHP native types will be converted to variants using the same rules as found in the constructor for the VARIANT class. COM and DOTNET objects will have the value of their default property taken and used as the variant value.

The variant arithmetic functions are wrappers around the similarly named functions in the COM library; for more information on these functions, consult the MSDN library. The PHP functions are named slightly differently; for example variant_add() in PHP corresponds to VarAdd() in the MSDN documentation.

variant_set_type

(PHP 5)

variant_set_type -- Convert a variant into another type "in-place"

Description

void **variant_set_type** (variant variant, int type)

This function is similar to **variant_cast()** except that the variant is modified "in-place"; no new variant is created. The parameters for this function have identical meaning to those of **variant_cast()**.

See also **variant_cast()**.

variant_set

(PHP 5)

variant_set -- Assigns a new value for a variant object

Description

void **variant_set** (variant variant, mixed value)

Converts *value* to a variant and assigns it to the *variant* object; no new variant object is created, and the old value of *variant* is freed/released.

variant_sub

(PHP 5)

variant_sub -- subtracts the value of the right variant from the left variant value and returns the result

Description

mixed **variant_sub** (mixed left, mixed right)

Subtracts *right* from *left* using the following rules:

Tabla 1. Variant Subtraction Rules

If	Then
Both expressions are of the string type	Subtraction
One expression is a string type and the other a character	Subtraction
One expression is numeric and the other is a string	Subtraction.
Both expressions are numeric	Subtraction
Either expression is NULL	NULL is returned
Both expressions are empty	Empty string is returned

See also **variant_add()**.

> **Nota:** As with all the variant arithmetic functions, the parameters for this function can be either a PHP native type (integer, string, floating point, boolean or **NULL**), or an instance of a COM, VARIANT or DOTNET class. PHP native types will be converted to variants using the same rules as found in the constructor for the VARIANT class. COM and DOTNET objects will have the value of their default property taken and used as the variant value.
>
> The variant arithmetic functions are wrappers around the similarly named functions in the COM library; for more information on these functions, consult the MSDN library. The PHP functions are named slightly differently; for example variant_add() in PHP corresponds to VarAdd() in the MSDN documentation.

variant_xor

(PHP 5)

variant_xor -- Performs a logical exclusion on two variants

Description

mixed **variant_xor** (mixed left, mixed right)

Performs a logical exclusion, according to the following truth table:

Tabla 1. Variant XOR Rules

If *left* is	If *right* is	then the result is
TRUE	**TRUE**	**FALSE**

If *left* is	If *right* is	then the result is
TRUE	**FALSE**	**TRUE**
FALSE	**TRUE**	**TRUE**
FALSE	**FALSE**	**FALSE**
NULL	**NULL**	**NULL**

See also **variant_and()**, **variant_or()**.

> **Nota:** As with all the variant arithmetic functions, the parameters for this function can be either a PHP native type (integer, string, floating point, boolean or **NULL**), or an instance of a COM, VARIANT or DOTNET class. PHP native types will be converted to variants using the same rules as found in the constructor for the VARIANT class. COM and DOTNET objects will have the value of their default property taken and used as the variant value.
>
> The variant arithmetic functions are wrappers around the similarly named functions in the COM library; for more information on these functions, consult the MSDN library. The PHP functions are named slightly differently; for example variant_add() in PHP corresponds to VarAdd() in the MSDN documentation.

XIII. Funciones ClibPDF

Introducción

ClibPDF le permite crear documentos PDF con PHP. La funcionalidad y la interfaz de programación de ClibPDF son similares a PDFlib. Esta documentación debería ser leído junto con el manual de ClibPDF, ya que éste explica la biblioteca con mucho mayor detalle.

Muchas funciones en la biblioteca ClibPDF nativa y el módulo PHP, así como en PDFlib, tienen el mismo nombre. Todas las funciones excepto por **cpdf_open()** reciben el gestor del documento como su primer parámetro.

Actualmente este gestor no es usado internamente ya que ClibPDF no soporta la creación de varios documentos PDF a la vez. De hecho, no debería intentarlo siquiera, los resultados son impredecibles. Es difícil hacerse una idea de las consecuencias que representaría en un entorno multi-hilos. De acuerdo al autor de ClibPDF, esto cambiará en uno de los lanzamientos siguientes (la versión actual cuando se escribieron éstas líneas es 1.10). Si necesita esta funcionalidad, use el módulo pdflib.

Una característica interesante de ClibPDF (y PDFlib) es la habilidad de crear el documento pdf completamente en memoria sin usar archivos temporales. También provee la habilidad de pasar coordenadas en una unidad de longitud predefinida. (Esta característica puede ser simulada también por **pdf_translate()** cuando se usan las funciones PDFlib.)

Otra característica interesante de ClibPDF es el hecho de que cualquier página puede ser modificada en cualquier momento, incluso si una nueva página ya ha sido abierta. La función **cpdf_set_current_page()** permite abandonar la página actual y posiblemente modificar otra página.

La mayoría de funciones son razonablemente fáciles de usar. La parte más difícil es probablemente la creación misma de un documento PDF muy sencillo. El ejemplo presentado a continuación debería ayudarlo a iniciar. En éste se crea un documento con una página. La página contiene el texto

"Times-Roman" en una fuente tipográfica de borde exterior y 30pt. El texto es subrayado.

> **Nota:** Si está interesado en generadores de PDF gratuitos alternativos que no usen bibliotecas PDF externas, vea este FAQ relacionado.

Requirimientos

Para poder usar las funciones ClibPDF necesita instalar el paquete ClibPDF. Éste se encuentra disponible para su descarga en FastIO, pero requiere que compre una licencia para su uso comercial. PHP requiere que usted use cpdflib >= 2.

Instalación

To get these functions to work, you have to compile PHP with *--with-cpdflib[=DIR]*. DIR is the cpdflib install directory, defaults to `/usr`. In addition you can specify the jpeg library and the tiff library for ClibPDF to use. To do so add to your configure line the options *--with-jpeg-dir[=DIR] --with-tiff-dir[=DIR]*.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

`CPDF_PM_NONE` (**integer**)

`CPDF_PM_OUTLINES` (**integer**)

`CPDF_PM_THUMBS` (**integer**)

`CPDF_PM_FULLSCREEN` (**integer**)

`CPDF_PL_SINGLE` (**integer**)

`CPDF_PL_1COLUMN` (**integer**)

`CPDF_PL_2LCOLUMN` (**integer**)

`CPDF_PL_2RCOLUMN` (**integer**)

Ejemplos

Ejemplo 1. Ejemplo Simple de ClibPDF

```
<?php
$cpdf = cpdf_open(0);
cpdf_page_init($cpdf, 1, 0, 595, 842, 1.0);
cpdf_add_outline($cpdf, 0, 0, 0, 1, "Pag. 1");
cpdf_begin_text($cpdf);
cpdf_set_font($cpdf, "Times-Roman", 30, "WinAnsiEncoding");
cpdf_set_text_rendering($cpdf, 1);
cpdf_text($cpdf, "Times Roman outlined", 50, 50);
cpdf_end_text($cpdf);
cpdf_moveto($cpdf, 50, 50);
cpdf_lineto($cpdf, 740, 330);
cpdf_stroke($cpdf);
cpdf_finalize_page($cpdf, 1);
cpdf_finalize($cpdf);
Header("Content-type: application/pdf");
cpdf_output_buffer($cpdf);
cpdf_close($cpdf);
?>
```

La distribución de pdflib contiene un ejemplo más complejo que genera una serie de páginas con un reloj análogo. Aquí está ese ejemplo convertido a PHP usando la extensión ClibPDF:

Ejemplo 2. Ejemplo pdfclock tomado de la distribución de pdflib 2.0

```
<?php
$radio = 200;
$margen = 20;
$conteo_paginas = 40;

$pdf = cpdf_open(0);
cpdf_set_creator($pdf, "pdf_clock.php");
cpdf_set_title($pdf, "Reloj An&aacute;logo");

while ($conteo_paginas-- > 0) {
  cpdf_page_init($pdf, $conteo_paginas+1, 0, 2 * ($radio + $margen), 2 * ($radio + $mar

  cpdf_set_page_animation($pdf, 4, 0.5, 0, 0, 0);  /* limpiar */

  cpdf_translate($pdf, $radio + $margen, $radio + $margen);
  cpdf_save($pdf);
  cpdf_setrgbcolor($pdf, 0.0, 0.0, 1.0);

  /* trazos de los minutos */
  cpdf_setlinewidth($pdf, 2.0);
  for ($alpha = 0; $alpha < 360; $alpha += 6) {
    cpdf_rotate($pdf, 6.0);
    cpdf_moveto($pdf, $radio, 0.0);
    cpdf_lineto($pdf, $radio-$margen/3, 0.0);
    cpdf_stroke($pdf);
  }

  cpdf_restore($pdf);
  cpdf_save($pdf);

  /* trazos de 5 minutos */
  cpdf_setlinewidth($pdf, 3.0);
  for ($alpha = 0; $alpha < 360; $alpha += 30) {
    cpdf_rotate($pdf, 30.0);
    cpdf_moveto($pdf, $radio, 0.0);
    cpdf_lineto($pdf, $radio-$margen, 0.0);
    cpdf_stroke($pdf);
  }

  $ltime = getdate();

  /* dibujar nuestra mano */
  cpdf_save($pdf);
  cpdf_rotate($pdf, -(($ltime['minutes']/60.0) + $ltime['hours'] - 3.0) * 30.0);
  cpdf_moveto($pdf, -$radio/10, -$radio/20);
  cpdf_lineto($pdf, $radio/2, 0.0);
  cpdf_lineto($pdf, -$radio/10, $radio/20);
  cpdf_closepath($pdf);
  cpdf_fill($pdf);
  cpdf_restore($pdf);

  /* dibujar la mano de minutos */
  cpdf_save($pdf);
  cpdf_rotate($pdf, -(($ltime['seconds']/60.0) + $ltime['minutes'] - 15.0) * 6.0);
  cpdf_moveto($pdf, -$radio/10, -$radio/20);
  cpdf_lineto($pdf, $radio * 0.8, 0.0);
  cpdf_lineto($pdf, -$radio/10, $radio/20);
  cpdf_closepath($pdf);
  cpdf_fill($pdf);
  cpdf_restore($pdf);

  /* dibujar la mano de segundos */
  cpdf_setrgbcolor($pdf, 1.0, 0.0, 0.0);
  cpdf_setlinewidth($pdf, 2);
  cpdf_save($pdf);
  cpdf_rotate($pdf, -(($ltime['seconds'] - 15.0) * 6.0));
  cpdf_moveto($pdf, -$radio/5, 0.0);
  cpdf_lineto($pdf, $radio, 0.0);
  cpdf_stroke($pdf);
  cpdf_restore($pdf);

  /* dibujar un pequenyo circulo en el centro */
```

Ver también

Vea también la documentación de la extensión PDFlib.

Tabla de contenidos

cpdf_set_text_matrix -- Define la matriz de texto
cpdf_set_text_pos -- Define la posición del texto
cpdf_set_text_rendering -- Determina cómo es presentado el texto
cpdf_set_text_rise -- Define la elevación del texto
cpdf_set_title -- Define el campo title de un documento PDF
cpdf_set_viewer_preferences -- How to show the document in the viewer
cpdf_set_word_spacing -- Define el espacio entre palabras
cpdf_setdash -- Defina el patrón de la raya
cpdf_setflat -- Define la monotonía
cpdf_setgray_fill -- Pone el color de relleno al valor gris
cpdf_setgray_stroke -- Define el color para dibujar al valor gris
cpdf_setgray -- Pone el color de relleno y dibujo a gris
cpdf_setlinecap -- Define el parámetro linecap
cpdf_setlinejoin -- Define el parámetro linejoin
cpdf_setlinewidth -- Define la profundidad de la línea
cpdf_setmiterlimit -- Define el límite del inglete
cpdf_setrgbcolor_fill -- Pone el color de relleno a l valor de clor rgb
cpdf_setrgbcolor_stroke -- Pone el color de dibujo al valor de color rgb
cpdf_setrgbcolor -- Pone el color de relleno y dibujo al valor de color rgb
cpdf_show_xy -- Muestra texto en la posición
cpdf_show -- Muestra el texto en la posición actual
cpdf_stringwidth -- Devuelve la anchura del texto en la fuente actual
cpdf_stroke -- Dibuja una línea a lo largo del camino
cpdf_text -- Muestra texto conparámetros
cpdf_translate -- Define el sistema de origen de coordenadas

cpdf_add_annotation

(PHP 3>= 3.0.12, PHP 4 , PHP 5)

cpdf_add_annotation -- Añade una anotación

Descripción

void **cpdf_add_annotation** (int pdf document, double llx, double lly, double urx, double ury, string title, string content, int mode)

La función **cpdf_add_annotation()** añade una nota con la esquina inferior izquierda en (*llx*, *lly*) y la esquina superior derecha en (*urx*, *ury*).

El últomo parámetro opcional determina el tamaño de la unidad. Si es 0 o se omite, se usa la unidad por defecto especificada para la página. De otro modo las coordenadas se miden en puntos postscript, despreciando la unidad actual.

cpdf_add_outline

(PHP 3>= 3.0.9, PHP 4 , PHP 5)

cpdf_add_outline -- Añade una marca en la página actual

Descripción

void **cpdf_add_outline** (int pdf document, string text)

La función **cpdf_add_outline()** añade una marca con el texto *text* que apunta a la página actual.
Ejemplo 1. Añadiendo un contorno de página

```
<?php
$cpdf = cpdf_open(0);
cpdf_page_init($cpdf, 1, 0, 595, 842);
cpdf_add_outline($cpdf, 0, 0, 0, 1, "Página 1");
// ...
// Algún dibujo
// ...
cpdf_finalize($cpdf);
Header("Content-type: application/pdf");
cpdf_output_buffer($cpdf);
cpdf_close($cpdf);
?>
```

cpdf_arc

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_arc -- Dibuja un arco

Descripión

void **cpdf_arc** (int pdf document, double x-koor, double y-koor, double radius, double start, double end, int mode)

La función **cpdf_arc()** dibuja un arco con el centro rn el punto (*x-koor*, *y-koor*) y radio *radius*, empezando en el ángulo *start* y terminando en el ángulo *end*.

El último parámetro opcional especifica el tamaño de la unidad. Si es 0 o se omite, se usa la unidad especificada por defecto. De otro modo las coordenadas son medidas en puntos postscript,despreciando la unidad actual.

Vea también **cpdf_circle()**.

cpdf_begin_text

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_begin_text -- Inicializa una sección de texto

Descripción

void **cpdf_begin_text** (int pdf document)

La función **cpdf_begin_text()** comienza una sección de texto. Debe ser terminada con **cpdf_end_text()**.
Ejemplo 1. Salida de texto

```
<?php cpdf_begin_text($pdf);
cpdf_set_font($pdf, 16, "Helvetica", "WinAnsiEncoding");
cpdf_text($pdf, 100, 100, "Algún texto");
cpdf_end_text($pdf) ?>
```

Vea también **cpdf_end_text()**.

cpdf_circle

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_circle -- Dibuja un círculo

Descripción

void **cpdf_circle** (int pdf document, double x-koor, double y-koor, double radius, int mode)

La función **cpdf_circle()** dibuja un círculo con centro en el punto (*x-koor*, *y-koor*) y radio *radius*.

El último parámetro opcional define el tamaño de la unidad. Si es 0 o se omite, se usa el valor por defecto para la página. De otro modo las coordenadas se miden en puntos postscript, despreciando la unidad actual.

Vea también **cpdf_arc()**.

cpdf_clip

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_clip -- Ajusta al camino actual

Descripción

void **cpdf_clip** (int pdf document)

La función **cpdf_clip()** ajusta todos los dibujos al camino actual.

cpdf_close

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_close -- Cierra un documento PDF

Descripción

void **cpdf_close** (int pdf document)

La función **cpdf_close()** cierra un documento PDF. Esta debería ser la última operación incluso después de **cpdf_finalize()**, **cpdf_output_buffer()** y **cpdf_save_to_file()**.

Vea también **cpdf_open()**.

cpdf_closepath_fill_stroke

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_closepath_fill_stroke -- Cierra, llena y traza el camino actual

Descripción

void **cpdf_closepath_fill_stroke** (int pdf document)

La función **cpdf_closepath_fill_stroke()** cierra, llena el interior del caminoa catual con el color actual de relleno y dibuja el camino actual.

Vea también **cpdf_closepath()**, **cpdf_stroke()**, **cpdf_fill()**, **cpdf_setgray_fill()**, **cpdf_setgray()**, **cpdf_setrgbcolor_fill()**, **cpdf_setrgbcolor()**.

cpdf_closepath_stroke

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_closepath_stroke -- Cierra el camino y dibuja una línea a lo largo del camino

Descripción

void **cpdf_closepath_stroke** (int pdf document)

La función **cpdf_closepath_stroke()** es una combinación de **cpdf_closepath()** y **cpdf_stroke()**. Después limpia el camino.

Vea también **cpdf_closepath()**, **cpdf_stroke()**.

cpdf_closepath

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_closepath -- Cierra el camino

Descripción

void **cpdf_closepath** (int pdf document)

La función **cpdf_closepath()** cierra el camino actual.

cpdf_continue_text

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_continue_text -- Pone texto en la línea siguiente

Descripción

void **cpdf_continue_text** (int pdf document, string text)

La función **cpdf_continue_text()** pone la cadena *text* en la línea siguiente.

Vea también **cpdf_show_xy()**, **cpdf_text()**, **cpdf_set_leading()**, **cpdf_set_text_pos()**.

cpdf_curveto

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_curveto -- Dibuja una curva

Descripción

void **cpdf_curveto** (int pdf document, double x1, double y1, double x2, double y2, double x3, double y3, int mode)

La función **cpdf_curveto()** dibuja una curva Bezier desde el punto actual al punto (*x3*, *y3*) usando (*x1*, *y1*) y (*x2*, *y2*) como puntos de control.

El último parámetro opcional especifica el tamaño de la unidad. Si es 0 o se omite, se usa la unidad especificada para la página. De otro modo las coordenadas se miden en puntos postscript, despreciando la unidad en curso.

Vea también **cpdf_moveto()**, **cpdf_rmoveto()**, **cpdf_rlineto()**, **cpdf_lineto()**.

cpdf_end_text

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_end_text -- Finaliza una sección de texto

Descripción

void **cpdf_end_text** (int pdf document)

La función **cpdf_end_text()** finaliza unasección de texto que fue inicializada con **cpdf_begin_text()**.

Ejemplo 1. Salida de texto

```
<?php cpdf_begin_text($pdf);
cpdf_set_font($pdf, 16, "Helvetica", "WinAnsiEncoding");
cpdf_text($pdf, 100, 100, "Algún texto");
cpdf_end_text($pdf) ?>
```

Vea también **cpdf_begin_text()**.

cpdf_fill_stroke

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_fill_stroke -- LLena y traza el camino actual

Descripción

void **cpdf_fill_stroke** (int pdf document)

La función **cpdf_fill_stroke()** llena el interior del camino actual con el color de relleno actual y dibuja el camino actual.

Vea también **cpdf_closepath()**, **cpdf_stroke()**, **cpdf_fill()**, **cpdf_setgray_fill()**, **cpdf_setgray()**, **cpdf_setrgbcolor_fill()**, **cpdf_setrgbcolor()**.

cpdf_fill

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_fill -- LLena el camino actual

Descripción

void **cpdf_fill** (int pdf document)

La función **cpdf_fill()** llena el interior del camino actual con el color alctual de relleno.

Vea también **cpdf_closepath()**, **cpdf_stroke()**, **cpdf_setgray_fill()**, **cpdf_setgray()**, **cpdf_setrgbcolor_fill()**, **cpdf_setrgbcolor()**.

cpdf_finalize_page

(PHP 3>= 3.0.10, PHP 4 , PHP 5)

cpdf_finalize_page -- Finaliza una página

Descripción

void **cpdf_finalize_page** (int pdf document, int page number)

La función **cpdf_finalize_page()** finaliza una página con número de página *page number*. Esta función es sólo para ahorrar memoria. Una página terminada ocupa menos memoria pero no puede volver a ser modificada.

Vea también **cpdf_page_init()**.

cpdf_finalize

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_finalize -- Finaliza un documento

Descripción

void **cpdf_finalize** (int pdf document)

La función **cpdf_finalize()** finaliza un documento. Aún se tiene que llamar a **cpdf_close()**.

Vea también **cpdf_close()**.

cpdf_global_set_document_limits

(PHP 4 , PHP 5)

cpdf_global_set_document_limits -- Sets document limits for any pdf document

Description

void **cpdf_global_set_document_limits** (int maxpages, int maxfonts, int maximages, int maxannotations, int maxobjects)

La función **cpdf_global_set_document_limits()** define varios límites del documento. Esta función debe ser llamada antes de **cpdf_open()** para que haga efecto. Ello define los límites de cualquier documento abierto con anterioridad.

Vea también **cpdf_open()**.

cpdf_import_jpeg

(PHP 3>= 3.0.9, PHP 4 , PHP 5)

cpdf_import_jpeg -- Abre una imagen JPEG

Descripción

int **cpdf_import_jpeg** (int pdf document, string file name, double x-koor, double y-koor, double angle, double width, double height, double x-scale, double y-scale, int mode)

La función **cpdf_import_jpeg()** abre una imagen almacenada en el fichero de nombre *file name*. El formato de la imagen debe ser JPEG. La imagen es situada en la página actual en la posición (*x-koor*, *y-koor*). La imagen es rotada *angle* grados.

El último parámetro opcional determina el tamaño de la unidad. Si es 0 o se omite, se usa la unidad por defecto especificada para la página. De otro modo las coordenadas se miden en puntos postscript, despreciando la unidad actual.

Vea también **cpdf_place_inline_image()**,

cpdf_lineto

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_lineto -- Dibuja una línea

Descripción

void **cpdf_lineto** (int pdf document, double x-koor, double y-koor, int mode)

La función **cpdf_lineto()** dibuja una línea desde el punto actual al punto con coordenadas (*x-koor*, *y-koor*).

El último parámetro opcional determina el tamaño de la unidad. Si es 0 o se omite, se usa el valor especificado para la página por defecto. De otro modo las coordenadas se miden en puntos postscript, despreciando la unidad actual.

Vea también **cpdf_moveto()**, **cpdf_rmoveto()**, **cpdf_curveto()**.

cpdf_moveto

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_moveto -- Define el punto actual

Descripción

void **cpdf_moveto** (int pdf document, double x-koor, double y-koor, int mode)

La función**cpdf_moveto()** pone el punto actual en las coordenadas *x-koor* y *y-koor*.

El último parámetro opcional determina la longitud de la unidad. Si es 0 o se omite, la unidad por defecto será la especificada para la página. De otro modo las coordenadas se medirán en puntos postscript despreciando la unidad en curso.

cpdf_newpath

(PHP 3>= 3.0.9, PHP 4 , PHP 5)

cpdf_newpath -- Starts a new path

Description

bool **cpdf_newpath** (int pdf_document)

The **cpdf_newpath()** starts a new path on the document given by the *pdf_document* parameter. Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

cpdf_open

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_open -- Abre un nuevo documento pdf

Descripción

int **cpdf_open** (int compresion [, string nombre_archivo [, array limites_doc]])

La función **cpdf_open()** abre un nuevo documento pdf. El primer parámetro habilita la compresión del documento si su valor es diferente de 0. El segundo parámetro opcional indica el archivo en el que es escrito el documento. Si se omite, el documento es creado en memoria, y puede ser escrito ya sea por medio de **cpdf_save_to_file()** o dirigido a la salida estándar con **cpdf_output_buffer()**.

> **Nota:** El valor de retorno será necesario en versiones posteriores de ClibPDF como el primer argumento en todas aquellas funciones que escriben sobre el documento pdf.
>
> La biblioteca ClibPDF toma el nombre de archivo "-" como un sinónimo de stdout. Si PHP es compilado como un módulo de apache, esto no funcionará debido a que el modo en que ClibPDF dirige su salida a stdout no funciona con apache. Puede resolver este problema evitando el uso del nombre de archivo, y usando **cpdf_output_buffer()** para volcar el documento pdf.

Vea también **cpdf_close()** y **cpdf_output_buffer()**.

cpdf_output_buffer

(PHP 3>= 3.0.9, PHP 4 , PHP 5)

cpdf_output_buffer -- Pone el documento PDF en el buffer de memoria

Descripción

void **cpdf_output_buffer** (int pdf document)

La función **cpdf_output_buffer()** muestra el documento PDF por la salida estándar. El documento debe ser creado en memoria, que es el caso de la función **cpdf_open()** cuando ha sido llamada sin parámetros.

Vea también **cpdf_open()**.

cpdf_page_init

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_page_init -- Comienza una nueva página

Descripción

void **cpdf_page_init** (int pdf document, int page number, int orientation, double height, double width, double unit)

La función **cpdf_page_init()** crea una nueva página de altura *height* y profundidad *width*. La página tiene el número *page number* y orientación *orientation*. *orientation* puede ser 0 para retrato y 1 para paisaje. El último parámetro opcional *unit* define la unidad del sistema de coordenadas. El valor debería ser el número de puntos postscript por unidad. Como el valor de una pulgada el igual a 72

puntos, un valor de 72 sería la unidad para una pulgada. Por defecto es 72.

Vea también **cpdf_set_current_page()**.

cpdf_place_inline_image

(PHP 3>= 3.0.9, PHP 4 , PHP 5)

cpdf_place_inline_image -- Situa una imagen en la página

Descripción

void **cpdf_place_inline_image** (int pdf document, int image, double x-koor, double y-koor, double angle, double width, double height, int mode)

La función **cpdf_place_inline_image()** situa una imagen creada con las funciones de imagenes de PHP en la posición de la página (*x-koor*, *y-koor*). La imagen puede ser escalada al mismo tiempo.

El último parámetro opcional determina el tamaño de la unidad. Si es 0 o se omite, se usa la unidad por defecto especificada para la página. De otro modo las coordenadas son medidas en puntos postscript, descartando la unidad actual.

Vea también **cpdf_import_jpeg()**,

cpdf_rect

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_rect -- Dibuja un rectángulo

Descripción

void **cpdf_rect** (int pdf document, double x-koor, double y-koor, double width, double height, int mode)

La función **cpdf_rect()** dibuja un rectángulo con su esquina inferior izquierda en el punto (*x-koor*, *y-koor*). La anchura es *widgth*. La altura es *height*.

El último parámetro opcional define el tamaño de la unidad. Si es 0 o se omite, se usa la unidad por defecto especificada para la página. De otro modo las coordenadas se miden en puntos postscript, despreciando la unidad actual.

cpdf_restore

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_restore -- Restaura un entorno formalmente salvado

Descripción

void **cpdf_restore** (int pdf document)

La función **cpdf_restore()** restaura el entorno salvado con **cpdf_save()**. Funciona como el comando grestore de postscript. Muy útil si se quiere trasladar o rotar un objeto sin afectar ortros objetos.

Ejemplo 1. Salvar/Restaurar

```
<?php cpdf_save($pdf);
// hacer todo tipo de rotaciones, transformaciones, ...
cpdf_restore($pdf) ?>
```

Vea también **cpdf_save()**.

cpdf_rlineto

(PHP 3>= 3.0.9, PHP 4 , PHP 5)

cpdf_rlineto -- Dibuja una línea

Descripción

void **cpdf_rlineto** (int pdf document, double x-koor, double y-koor, int mode)

La función **cpdf_rlineto()** dibuja una línea desde el punto actual al punto relativo con coordenadas (*x-koor*, *y-koor*).

El último parámetro opcional determina la longitud de la unidad. Si es 0 o se omite, se usa el valor por defecto para la página. De otro modo las coordenadas se miden en puntos postscript, despreciando la unidad actual.

Vea también **cpdf_moveto()**, **cpdf_rmoveto()**, **cpdf_curveto()**.

cpdf_rmoveto

(PHP 3>= 3.0.9, PHP 4 , PHP 5)

cpdf_rmoveto -- Define el punto actual

Descripción

void **cpdf_rmoveto** (int pdf document, double x-koor, double y-koor, int mode)

La función **cpdf_rmoveto()** pone el punto actual relativo a las coordenadas *x-koor* y *y-koor*.

El último parámetro opciona determina la loingitud de la unidad. Si es 0 o se omite, la unidad por defecto será la especificada para la página. De otro modo las coordenadas se medirán en puntos postscript, despreciando la unidad en curso.

Vea también **cpdf_moveto()**.

cpdf_rotate_text

(PHP 3>= 3.0.9, PHP 4 , PHP 5)

cpdf_rotate_text -- Sets text rotation angle

Description

bool **cpdf_rotate_text** (int pdfdoc, float angle)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

cpdf_rotate

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_rotate -- Define la rotación

Descripción

void **cpdf_rotate** (int pdf document, double angle)

La función **cpdf_rotate()** define la rotación en *angle* grados.

cpdf_save_to_file

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_save_to_file -- Escribe el documento PDF en un fichero

Descripción

void **cpdf_save_to_file** (int pdf document, string filename)

La función **cpdf_save_to_file()** guarda el documento PDF en un fichero si este documeto ha sido creado en memoria. Esta función no es necesaria si el documento PDF ha sido abierto mediante la especificación de un nombre de fichero en la función **cpdf_open()**.

Vea también **cpdf_output_buffer()**, **cpdf_open()**.

cpdf_save

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_save -- Salva el entorno actual

Descripción

void **cpdf_save** (int pdf document)

La función **cpdf_save()** salva el entorno actual. Funciona como el comando gsave de postscript. Muy útil si se quiere trasladar o trotar un objeto sin afetar a los demás.

Vea también **cpdf_restore()**.

cpdf_scale

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_scale -- Define la escala

Descripción

void **cpdf_scale** (int pdf document, double x-scale, double y-scale)

La función **cpdf_scale()** define el factor de escala en los dos sentidos.

cpdf_set_action_url

(PHP 3>= 3.0.9, PHP 4 , PHP 5)

cpdf_set_action_url -- Sets hyperlink

Description

bool **cpdf_set_action_url** (int pdfdoc, float xll, float yll, float xur, float xur, string url [, int mode])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

cpdf_set_char_spacing

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_set_char_spacing -- Determina el espacio entre caracteres

Descripción

void **cpdf_set_char_spacing** (int pdf document, double space)

LA función **cpdf_set_char_spacing()** define el espacio entre caracteres.

Vea también **cpdf_set_word_spacing()**, **cpdf_set_leading()**.

cpdf_set_creator

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_set_creator -- Define el campo creator en el documento PDF

Descripción

void **cpdf_set_creator** (string creator)

La función **cpdf_set_creator()** define el creador de un documento PDF.

Vea también **cpdf_set_subject()**, **cpdf_set_title()**, **cpdf_set_keywords()**.

cpdf_set_current_page

(PHP 3>= 3.0.9, PHP 4 , PHP 5)

cpdf_set_current_page -- Define la página actual

Descripción

void **cpdf_set_current_page** (int pdf document, int page number)

La función **cpdf_set_current_page()** define la página en la que se van a realizar todas las operaciones. Uno puede cambiar entre páginas a menos que una página ha sido finalizada con **cpdf_finalize_page()**.

Vea también **cpdf_finalize_page()**.

cpdf_set_font_directories

(PHP 4 >= 4.0.6, PHP 5)

cpdf_set_font_directories -- Sets directories to search when using external fonts

Description

bool **cpdf_set_font_directories** (int pdfdoc, string pfmdir, string pfbdir)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

cpdf_set_font_map_file

(PHP 4 >= 4.0.6, PHP 5)

cpdf_set_font_map_file -- Sets fontname to filename translation map when using external fonts

Description

bool **cpdf_set_font_map_file** (int pdfdoc, string filename)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

cpdf_set_font

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_set_font -- Selecciona la fuente y el tamaño actual

Descripción

void **cpdf_set_font** (int pdf document, string font name, double size, string encoding)

La función **cpdf_set_font()** define la fuente actual, el tamaño y la codificación. Actualmente solo son soportadas las fuentes estándar de postscript. El último parámetro *encoding* puede tomar los siguientes valores: "MacRomanEncoding", "MacExpertEncoding", "WinAnsiEncoding", y "**NULL**". "**NULL**" es para el cifrado incluído en la fuente. Para mas información vea el manual de ClibPDF, especialmente para cómo soportar las fuentes asiáticas.

cpdf_set_horiz_scaling

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_set_horiz_scaling -- Define la escala horizontal del texto

Descripción

void **cpdf_set_horiz_scaling** (int pdf document, double scale)

La función **cpdf_set_horiz_scaling()** define la escala horizontal al *scale* por ciento.

cpdf_set_keywords

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_set_keywords -- Pone el valor del campo 'keywords'(palabras clave) de un documento PDF

Descripción

void **cpdf_set_keywords** (string keywords)

La función **cpdf_set_keywords()** define las palabras clave de un documento PDF.

Vea también **cpdf_set_title()**, **cpdf_set_creator()**, **cpdf_set_subject()**.

cpdf_set_leading

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_set_leading -- Define la distancias entre las líneas de texto

Descripción

void **cpdf_set_leading** (int pdf document, double distance)

La función **cpdf_set_leading()** define la distancia entre las líneas de texto. Esto se usará si el texto es la salida de **cpdf_continue_text()**.

Vea también **cpdf_continue_text()**.

cpdf_set_page_animation

(PHP 3>= 3.0.9, PHP 4 , PHP 5)

cpdf_set_page_animation -- Define la separación entre páginas

Descripción

void **cpdf_set_page_animation** (int pdf document, int transition, double duration)

La función **cpdf_set_page_animation()** define la transición entre páginas que se siguen.

El valor de *transition* puede ser

0 para ninguno,
1 para dos líneas que se barren a través de la pantalla, revelen la página,
2 para múltiples líneas,
3 para que una caja revele la página,
4 para una única línea,
5 para que la página naterior se disipe para revelar la pagina,
6 para que el efecto de disolución se mueva de un extremop de la página al otro,
7 para que la página antígua simplemente sea reemplazada por la nueva página (default)
El valor de *duration* es el número de segundos entre las páginas que se pasan.

cpdf_set_subject

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_set_subject -- Define el valor del campo subjet de un documento PDF

Descripción

void **cpdf_set_subject** (string subject)

La función **cpdf_set_subject()** define el asunto de un documento PDF

Vea también **cpdf_set_title()**, **cpdf_set_creator()**, **cpdf_set_keywords()**.

cpdf_set_text_matrix

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_set_text_matrix -- Define la matriz de texto

Descripción

void **cpdf_set_text_matrix** (int pdf document, array matrix)

La función **cpdf_set_text_matrix()** define una matriz que describe una transformación aplicada a la fuente actual de texto.

cpdf_set_text_pos

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_set_text_pos -- Define la posición del texto

Descripción

void **cpdf_set_text_pos** (int pdf document, double x-koor, double y-koor, int mode)

La función **cpdf_set_text_pos()** define la posición del texto para la siguiente llamada a **cpdf_show ()**.

El último parámetro opcional *mode* determina la longitud de la unidad. Si es 0 o se omite, se usa el valor por defecto para la página. De otro modo, las coordenadas son medidas en puntos postscript, despreciando la unidad actual.

Vea también **cpdf_show()**, **cpdf_text()**.

cpdf_set_text_rendering

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_set_text_rendering -- Determina cómo es presentado el texto

Descripción

void **cpdf_set_text_rendering** (int pdf document, int mode)

La función **cpdf_set_text_rendering()** determina cómo es presentado el texto. Los posibles valores para *mode* son 0=llenar texto, 1=poner texto, 2=llenar y poner texto, 3=invisible, 4=llenar texto y añadirlo al camino de corte, 5=poner texto y añadirlo al camino de corte, 6=llenar y poner texto y añadirlo al camino de corte, 7=añadirlo al camino de corte

cpdf_set_text_rise

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_set_text_rise -- Define la elevación del texto

Descripción

void **cpdf_set_text_rise** (int pdf document, double value)

La función **cpdf_set_text_rise()** define la elevación del texto a *value* unidades.

cpdf_set_title

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_set_title -- Define el campo title de un documento PDF

Descripción

void **cpdf_set_title** (string title)

La función **cpdf_set_title()** define el título de un documento PDF

Vea también **cpdf_set_subject()**, **cpdf_set_creator()**, **cpdf_set_keywords()**.

cpdf_set_viewer_preferences

(PHP 3>= 3.0.9, PHP 4 , PHP 5)

cpdf_set_viewer_preferences -- How to show the document in the viewer

Description

bool **cpdf_set_viewer_preferences** (int pdfdoc, array preferences)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

cpdf_set_word_spacing

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_set_word_spacing -- Define el espacio entre palabras

Descripción

void **cpdf_set_word_spacing** (int pdf document, double space)

La función **cpdf_set_word_spacing()** especifica el espacio entre palabras.

Vea también **cpdf_set_char_spacing()**, **cpdf_set_leading()**.

cpdf_setdash

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_setdash -- Defina el patrón de la raya

Descripción

void **cpdf_setdash** (int pdf document, double white, double black)

La función **cpdf_setdash()** define el patrón de la raya *white* unidades blancas y *black* unidades negras. Si los dos son 0 se pone una línea sólida.

cpdf_setflat

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_setflat -- Define la monotonía

Descripción

void **cpdf_setflat** (int pdf document, double value)

La función **cpdf_setflat()** pone la monotonía a un valor de entre 0 y 100.

cpdf_setgray_fill

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_setgray_fill -- Pone el color de relleno al valor gris

Descripción

void **cpdf_setgray_fill** (int pdf document, double value)

La función **cpdf_setgray_fill()** define el valor de gris actual para rellelanr un camino.

Vea también **cpdf_setrgbcolor_fill()**.

cpdf_setgray_stroke

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_setgray_stroke -- Define el color para dibujar al valor gris

Descripción

void **cpdf_setgray_stroke** (int pdf document, double gray value)

La función **cpdf_setgray_stroke()** pone el color de dibujo actual al valor de gris dado.

Vea también **cpdf_setrgbcolor_stroke()**.

cpdf_setgray

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_setgray -- Pone el color de relleno y dibujo a gris

Descripción

void **cpdf_setgray** (int pdf document, double gray value)

La función **cpdf_setgray_stroke()** pone el color de relleno y dibujo al color gris dado.

Vea también **cpdf_setrgbcolor_stroke()**, **cpdf_setrgbcolor_fill()**.

cpdf_setlinecap

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_setlinecap -- Define el parámetro linecap

Description

void **cpdf_setlinecap** (int pdf document, int value)

La función **cpdf_setlinecap()** define el parámetro linecap entre los valores 0 y 2. 0 = empalmar al final, 1 = redondear, 2 = esquina proyectada

cpdf_setlinejoin

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_setlinejoin -- Define el parámetro linejoin

Descripción

void **cpdf_setlinejoin** (int pdf document, long value)

La función **cpdf_setlinejoin()** define el parámetro entre un valor de 0 y 2. 0 = ingletes, 1 = redondear, 2 = ángulo oblícuo

cpdf_setlinewidth

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_setlinewidth -- Define la profundidad de la línea

Descripción

void **cpdf_setlinewidth** (int pdf document, double width)

La función **cpdf_setlinewidth()** define la preofundidad de la línea a *width*.

cpdf_setmiterlimit

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_setmiterlimit -- Define el límite del inglete

Descripción

void **cpdf_setmiterlimit** (int pdf document, double value)

La función **cpdf_setmiterlimit()** define el límite del inglete a un valor mayor o igual a 1.

cpdf_setrgbcolor_fill

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_setrgbcolor_fill -- Pone el color de relleno a l valor de clor rgb

Descripción

void **cpdf_setrgbcolor_fill** (int pdf document, double red value, double green value, double blue value)

La función **cpdf_setrgbcolor_fill()** pone el color rgb actual para rellenar un camino.

Vea también **cpdf_setrgbcolor_stroke()**, **cpdf_setrgbcolor()**.

cpdf_setrgbcolor_stroke

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_setrgbcolor_stroke -- Pone el color de dibujo al valor de color rgb

Descripción

void **cpdf_setrgbcolor_stroke** (int pdf document, double red value, double green value, double blue value)

La función **cpdf_setrgbcolor_stroke()** pone el color de dibujo actual al valor de color rgb dado.

Vea también **cpdf_setrgbcolor_fill()**, **cpdf_setrgbcolor()**.

cpdf_setrgbcolor

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_setrgbcolor -- Pone el color de relleno y dibujo al valor de color rgb

Descripción

void **cpdf_setrgbcolor** (int pdf document, double red value, double green value, double blue value)

La función **cpdf_setrgbcolor_stroke()** pone el color de relleno y dibujo actual al color rgb dado.

Vea también **cpdf_setrgbcolor_stroke()**, **cpdf_setrgbcolor_fill()**.

cpdf_show_xy

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_show_xy -- Muestra texto en la posición

Descripción

void **cpdf_show_xy** (int pdf document, string text, double x-koor, double y-koor, int mode)

La función **cpdf_show_xy()** muestra la cadena *text* en la posición con coordenadas (*x-coor*, *y-coor*). El último parámetro opcional determina la longitud de la unidad. Si es 0 o se omite, se usa la unidad por defecto especificada para la página. De otro modo las coordenadas son medidas en puntos postscript, despreciando la unidad actual.

> **Nota:** La función **cpdf_show_xy()** es idéntica a **cpdf_text()** sin el parámetro opcional.

Vea también **cpdf_text()**.

cpdf_show

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_show -- Muestra el texto en la posición actual

Descripción

void **cpdf_show** (int pdf document, string text)

La función **cpdf_show()** muestra la cadena *text* en la posixción actual.

Vea también **cpdf_text()**, **cpdf_begin_text()**, **cpdf_end_text()**.

cpdf_stringwidth

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_stringwidth -- Devuelve la anchura del texto en la fuente actual

Descripción

double **cpdf_stringwidth** (int pdf document, string text)

La función **cpdf_stringwidth()** devuelve la anchura de la cadena *text*. Requiere haber definido antes una fuente.

Vea también **cpdf_set_font()**.

cpdf_stroke

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_stroke -- Dibuja una línea a lo largo del camino

Descripción

void **cpdf_stroke** (int pdf document)

La función **cpdf_stroke()** dibuja una línea a lo largo del camino actual.

Vea también **cpdf_closepath()**, **cpdf_closepath_stroke()**.

cpdf_text

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_text -- Muestra texto conparámetros

Descripción

void **cpdf_text** (int pdf document, string text, double x-koor, double y-koor, int mode, double orientation, int alignmode)

La función **cpdf_text()** muestra la cadena *text* en la posición de coordenadas (*x-coor*, *y-coor*). El parámero opcional determina la longitud de la unidad. Si es 0 o se omite, se usa la unidad por defecto especificada para la página. De otro modo las coordenadas son medidas en puntos postscript despreciando la unidad actual. El parámetro opcional *orientation* es la rotación del texto en grados. El parámetro opcional *alignmode* determina cómo está alineado el texto. Vea la documentación de ClibPDF para los posibles valores.

Vea también **cpdf_show_xy()**.

cpdf_translate

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

cpdf_translate -- Define el sistema de origen de coordenadas

Descripción

void **cpdf_translate** (int pdf document, double x-koor, double y-koor, int mode)

La función **cpdf_translate()** define el sistema origen de coordenadas en el punto (*x-coor*, *y-coor*).

El último parámetro opcional determina la longitud de la unidad. Si es 0 o se omite, se usa la unidad por defecto especificada en la página. De otro modo las coordenadas son medidas en puntos postscript, depreciando la unidad actual.

XIV. Crack Functions

Introducción

These functions allow you to use the CrackLib library to test the 'strength' of a password. The 'strength' of a password is tested by that checks length, use of upper and lower case and checked against the specified CrackLib dictionary. CrackLib will also give helpful diagnostic messages that will help 'strengthen' the password.

> **Nota:** This extension has been moved to the PECL repository and is no longer bundled with PHP as of PHP 5.0.0.

Requirimientos

More information regarding CrackLib along with the library can be found at http://www.crypticide.org/users/alecm/.

Instalación

Esta extension PECL no esta ligada a PHP. Mas informacion sobre nuevos lanzamientos, descargas ficheros de fuentes, informacion sobre los responsables asi como un 'CHANGELOG', se puede encontrar aqui: http://pecl.php.net/package/crack.

En PHP 4 la fuente de las extensiones PECL pueden encontrarse en el directorio `ext/` que se existe en las fuentes de PHP o en el enlace PECL de arriba. In order to use these functions you must compile PHP with Crack support by using the *--with-crack[=DIR]* configuration option.

Windows users will enable `php_crack.dll` inside of `php.ini` in order to use these functions. En PHP 4, esta DLL se encuentra en el directorio `extensions/` que existe en los binarios de PHP para Windows. Podeis descargar esta DLL de las extensiones PECL desde la pagina PHP Downloads o desde http://snaps.php.net/.

Configuración en tiempo de ejecución

El comportamiento de estas funciones está afectado por los valores definidos en `php.ini`.

Tabla 1. Crack configuration options

Name	Default	Changeable
crack.default_dictionary	NULL	PHP_INI_SYSTEM

For further details and definitions of the PHP_INI_* constants, see the Apéndice H.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Esta extensión no tiene ninguna constante definida.

Ejemplos

This example shows how to open a CrackLib dictionary, test a given password, retrieve any diagnostic messages, and close the dictionary.

Ejemplo 1. CrackLib example

```php
<?php
// Open CrackLib Dictionary
$dictionary = crack_opendict('/usr/local/lib/pw_dict')
     or die('Unable to open CrackLib dictionary');

// Perform password check
$check = crack_check($dictionary, 'gx9A2s0x');

// Retrieve messages
$diag = crack_getlastmessage();
echo $diag; // 'strong password'

// Close dictionary
crack_closedict($dictionary);
?>
```

Nota: If **crack_check()** returns **TRUE**, **crack_getlastmessage()** will return 'strong password'.

Tabla de contenidos

crack_check

(PHP 4 >= 4.0.5)

crack_check -- Performs an obscure check with the given password

Descripción

bool **crack_check** (resource dictionary, string password)

bool **crack_check** (string password)

Performs an obscure check with the given password on the specified dictionary.

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Lista de parámetros

dictionary

The crack lib dictionary. If not specified, the last opened dictionary is used.

password

The tested password.

Valores retornados

Returns **TRUE** if *password* is strong, or **FALSE** otherwise.

crack_closedict

(PHP 4 >= 4.0.5)

crack_closedict -- Closes an open CrackLib dictionary

Descripción

bool **crack_closedict** ([resource dictionary])

crack_closedict() closes the specified *dictionary* identifier.

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Lista de parámetros

dictionary

The dictionary to close. If not specified, the current directory is closed.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ver también

crack_opendict()

crack_getlastmessage

(PHP 4 >= 4.0.5)

crack_getlastmessage -- Returns the message from the last obscure check

Descripción

string **crack_getlastmessage** (void)

crack_getlastmessage() returns the message from the last obscure check.

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Valores retornados

The message from the last obscure check.

Ver también

crack_check()

crack_opendict

(PHP 4 >= 4.0.5)

crack_opendict -- Opens a new CrackLib dictionary

Descripción

resource **crack_opendict** (string dictionary)

crack_opendict() opens the specified CrackLib *dictionary* for use with **crack_check()**.

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Nota: Only one dictionary may be open at a time.

Lista de parámetros

dictionary

The path to the Cracklib dictionary.

Valores retornados

Returns a dictionary resource identifier on success, or **FALSE** on failure.

Ver también

crack_check()
crack_closedict()

XV. Funciones de Tipo de Caracter

Introducción

Las funciones que ofrece esta extensión chequean si un caracter o cadena se ubica dentro de una determinada clase de caracteres, de acuerdo a la localidad actual (vea también **setlocale()**).

Cuando son llamadas con un argumento entero, estas funciones se comportan exactamente como sus homólogos de C provenientes de `ctype.h`.

Cuando son llamadas con un argumento de tipo cadena, chequearán cada caracter de la cadena y solo devolverán **`TRUE`** si cada caracter de la cadena coincide con el criterio solicitado. Cuando son llamadas con una cadena vacía, el resultado será siempre **`TRUE`**.

Al pasar cualquier cosa diferente a una cadena o un entero, se devolverá **`FALSE`** inmediatamente.

Debe notarse que se prefieren siempre las funciones ctype sobre las expresiones regulares, e incluso sobre algunas funciones str_* e is_* ñalentes. Esto se debe a que ctype usa una biblioteca C nativa y por lo tanto realiza sus procesos significativamente más rápido.

Requirimientos

Ninguno aparte de las funciones de la biblioteca de C estándar, la cual se encuentra disponible siempre.

Instalación

A partir de PHP 4.2.0 estas funciones están habilitadas por defecto. En versiones anteriores es necesario configurar y compilar PHP con el parámetro *--enable-ctype*. Es posible deshabilitar el soporte para ctype con *--disable-ctype*.

La versión para Windows de *PHP* tiene soporte nativo para esta extensión. No se necesita cargar ninguna extensión adicional para usar estas funciones.

> **Nota:** A partir de PHP 4.3.0 el soporte para ctype es nativo.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Esta extensión no tiene ninguna constante definida.

Tabla de contenidos

ctype_alnum

(PHP 4 >= 4.0.4, PHP 5)

ctype_alnum -- Chequear posibles caracteres alfanuméricos

Descripción

bool **ctype_alnum** (string texto)

Devuelve **TRUE** si cada caracter del *texto* es o bien una letra o un dígito, o **FALSE** de lo contrario. En la localidad *C* estándar las letras se limitan a *[A-Za-z]* y la función es ñalente a *preg_match('/^[a-z0-9]*$/i', $texto)*.

Ejemplo 1. Un ejemplo de ctype_alnum() (usando la localidad predeterminada)

```
<?php
$cadenas = array('AbCd1zyZ9', 'foo!#$bar');
foreach ($cadenas as $caso_prueba) {
    if (ctype_alnum($caso_prueba)) {
        echo "La cadena $caso_prueba consiste completamente de letras o d&iacute;gitos.
    } else {
        echo "La cadena $caso_prueba no consiste completamente de letras o d&iacute;git
    }
}
?>
```

Este ejemplo producirá la salida:

```
La cadena AbCd1zyZ9 consiste completamente de letras o d&iacute;gitos.
La cadena foo!#$bar no consiste completamente de letras o d&iacute;gitos.
```

Vea también **ctype_alpha()**, **ctype_digit()**, y **setlocale()**.

ctype_alpha

(PHP 4 >= 4.0.4, PHP 5)

ctype_alpha -- Chequear posibles caracteres alfabéticos

Descripción

bool **ctype_alpha** (string texto)

Devuelve **TRUE** si cada caracter del *texto* es una letra de la localidad actual, o **FALSE** de lo contrario. En la localidad *C* estándar las letras se limitan a *[A-Za-z]* y **ctype_alpha()** es ñalente a *(ctype_upper($texto)* || *ctype_lower($texto))* si $texto es un caracter sencillo, aunque otros idiomas usan letras que no son consideradas como mayúsculas ni minúsculas.

Ejemplo 1. Un ejemplo de ctype_alpha() (usando la localidad predeterminada)

```
<?php
$cadenas = array('KjgWZC', 'arf12');
foreach ($cadenas as $caso_prueba) {
    if (ctype_alpha($caso_prueba)) {
        echo "La cadena $caso_prueba consiste completamente de letras.\n";
    } else {
        echo "La cadena $caso_prueba no consiste completamente de letras.\n";
    }
}
?>
```

Este ejemplo producirá la salida:

```
La cadena KjgWZC consiste completamente de letras.
La cadena arf12 no consiste completamente de letras.
```

Vea también **ctype_upper()**, **ctype_lower()**, y **setlocale()**.

ctype_cntrl

(PHP 4 >= 4.0.4, PHP 5)

ctype_cntrl -- Chequear posibles caracteres de control

Descripción

bool **ctype_cntrl** (string texto)

Devuelve **TRUE** si cada caracter del *texto* tiene una función especial de control, o **FALSE** de lo contrario. Los caracteres de control son, por ejemplo, la alimentación de línea, el tabulador, esc, etc.

Ejemplo 1. Un ejemplo de ctype_cntrl()

```
<?php
$cadenas = array('cadena1' => "\n\r\t", 'cadena2' => 'arf12');
foreach ($cadenas as $nombre => $caso_prueba) {
    if (ctype_cntrl($caso_prueba)) {
        echo "La cadena '$nombre' consiste completamente de caracteres de control.\n";
    } else {
        echo "La cadena '$nombre' no consiste completamente de caracteres de control.\n
    }
}
?>
```

Este ejemplo producirá la salida:

```
La cadena 'cadena1' consiste completamente de caracteres de control.
La cadena 'cadena2' no consiste completamente de caracteres de control.
```

ctype_digit

(PHP 4 >= 4.0.4, PHP 5)

ctype_digit -- Chequear posibles caracteres numéricos

Descripción

bool **ctype_digit** (string texto)

Devuelve **TRUE** si cada caracter del *texto* es un dígito decimal, o **FALSE** de lo contrario.

Ejemplo 1. Un ejemplo de ctype_digit()

```
<?php
$cadenas = array('1820.20', '10002', 'wsl!12');
foreach ($cadenas as $caso_prueba) {
    if (ctype_digit($caso_prueba)) {
        echo "La cadena $caso_prueba consiste completamente de d&iacute;gitos.\n";
    } else {
        echo "La cadena $caso_prueba no consiste completamente de d&iacute;gitos.\n";
    }
}
?>
```

Este ejemplo producirá la salida:

```
La cadena 1820.20 no consiste completamente de d&iacute;gitos.
La cadena 10002 consiste completamente de d&iacute;gitos.
La cadena wsl!12 no consiste completamente de d&iacute;gitos.
```

Vea también **ctype_alnum()** y **ctype_xdigit()**.

ctype_graph

(PHP 4 >= 4.0.4, PHP 5)

ctype_graph -- Chequear posibles caracteres imprimibles, con excepción de los espacios

Descripción

bool **ctype_graph** (string texto)

Devuelve **TRUE** si cada caracter del *texto* es imprimible y genera alguna salida visible (no incluye los espacios), o **FALSE** de lo contrario.

Ejemplo 1. Un ejemplo de ctype_graph()

```
<?php
$cadenas = array('cadena1' => "asdf\n\r\t", 'cadena2' => 'arf12', 'cadena3' => 'LKA#@%.
foreach ($cadenas as $nombre => $caso_prueba) {
    if (ctype_graph($caso_prueba)) {
        echo "La cadena '$nombre' consiste completamente de caracteres (visiblemente) i
    } else {
        echo "La cadena '$nombre' no consiste completamente de caracteres (visiblemente
    }
}
?>
```

Este ejemplo producirá la salida:

```
La cadena 'cadena1' no consiste completamente de caracteres (visiblemente) imprimibles.
La cadena 'cadena2' consiste completamente de caracteres (visiblemente) imprimibles.
La cadena 'cadena3' consiste completamente de caracteres (visiblemente) imprimibles.
```

Vea también **ctype_alnum()**, **ctype_print()**, y **ctype_punct()**.

ctype_lower

(PHP 4 >= 4.0.4, PHP 5)

ctype_lower -- Chequear posibles caracteres en minúscula

Descripción

bool **ctype_lower** (string texto)

Devuelve **TRUE** si cada caracter del *texto* es una letra minúscula en la localidad actual.

Ejemplo 1. Un ejemplo de ctype_lower() (usando la localidad predeterminada)

```
<?php
$cadenas = array('aac123', 'qiutoas', 'QASsdks');
foreach ($cadenas as $caso_prueba) {
    if (ctype_lower($caso_prueba)) {
        echo "La cadena $caso_prueba consiste completamente de letras min&uacute;sculas
    } else {
        echo "La cadena $caso_prueba no consiste completamente de letras min&uacute;scu
    }
}
?>
```

Este ejemplo producirá la salida:

```
La cadena aac123 no consiste completamente de letras min&uacute;sculas.
La cadena qiutoas consiste completamente de letras min&uacute;sculas.
La cadena QASsdks no consiste completamente de letras min&uacute;sculas.
```

Vea también **ctype_alpha()**, **ctype_upper()** y **setlocale()**.

ctype_print

(PHP 4 >= 4.0.4, PHP 5)

ctype_print -- Chequear posibles caracteres imprimibles

Descripción

bool **ctype_print** (string texto)

Devuelve **TRUE** si cada caracter del *texto* genera realmente alguna salida (incluyendo los espacios). Devuelve **FALSE** si el *texto* incluye caracteres de control o caracteres que no producen ninguna salida ni realizan función de control alguna después de todo.

Ejemplo 1. Un ejemplo de ctype_print()

```
<?php
$cadenas = array('cadena1' => "asdf\n\r\t", 'cadena2' => 'arf12', 'cadena3' => 'LKA#@%.
foreach ($cadenas as $nombre => $caso_prueba) {
    if (ctype_print($caso_prueba)) {
        echo "La cadena '$nombre' consiste completamente de caracteres imprimibles.\n";
    } else {
        echo "La cadena '$nombre' no consiste completamente de caracteres imprimibles.\
    }
}
?>
```

Este ejemplo producirá la salida:

```
La cadena 'cadena1' no consiste completamente de caracteres imprimibles.
La cadena 'cadena2' consiste completamente de caracteres imprimibles.
La cadena 'cadena3' consiste completamente de caracteres imprimibles.
```

Vea también **ctype_cntrl()**, **ctype_graph()**, y **ctype_punct()**.

ctype_punct

(PHP 4 >= 4.0.4, PHP 5)

ctype_punct -- Chequear posibles caracteres imprimibles que no son ni espacios en blanco ni caracteres alfanuméricos

Descripción

bool **ctype_punct** (string texto)

Devuelve **TRUE** si cada caracter del *texto* es imprimible, pero no es una letra, dígito o espacio en blanco; o **FALSE** de lo contrario.

Ejemplo 1. Un ejemplo de ctype_punct()

```
<?php
$cadenas = array('ABasdk!@!$#', '!@ # $', '*&$()');
foreach ($cadenas as $caso_prueba) {
    if (ctype_punct($caso_prueba)) {
        echo "La cadena $caso_prueba consiste completamente de signos de puntuaci&oacut
    } else {
        echo "La cadena $caso_prueba no consiste completamente de signos de puntuaci&oa
    }
}
?>
```

Este ejemplo producirá la salida:

```
La cadena ABasdk!@!$# no consiste completamente de signos de puntuaci&oacute;n.
La cadena !@ # $ no consiste completamente de signos de puntuaci&oacute;n.
La cadena *&$() consiste completamente de signos de puntuaci&oacute;n.
```

Vea también **ctype_cntrl()** y **ctype_graph()**.

ctype_space

(PHP 4 >= 4.0.4, PHP 5)

ctype_space -- Chequear posibles caracteres de espacio en blanco

Descripción

bool **ctype_space** (string texto)

Devuelve **TRUE** si cada caracter del *texto* genera cierto tipo de espacio en blanco, o **FALSE** de lo contrario. Junto con el caracter regular de espacio en blanco, también se consideran espacios a los caracteres de tabulación, tabulación vertical, alimentación de línea, retorno de carro y alimentación de formulario.

Ejemplo 1. Un ejemplo de ctype_space()

```
<?php
$cadenas = array('cadena1' => "\n\r\t", 'cadena2' => "\narf12", 'cadena3' => '\n\r\t');
foreach ($cadenas as $nombre => $caso_prueba) {
    if (ctype_space($caso_prueba)) {
        echo "La cadena '$nombre' consiste completamente de espacios en blanco.\n";
    } else {
        echo "La cadena '$nombre' no consiste completamente de espacios en blanco.\n";
    }
}
?>
```

Este ejemplo producirá la salida:

```
La cadena 'cadena1' consiste completamente de espacios en blanco.
La cadena 'cadena2' no consiste completamente de espacios en blanco.
La cadena 'cadena3' no consiste completamente de espacios en blanco.
```

Vea también **ctype_cntrl()**, **ctype_graph()**, y **ctype_punct()**.

ctype_upper

(PHP 4 >= 4.0.4, PHP 5)

ctype_upper -- Chequear posibles caracteres en mayúscula

Descripción

bool **ctype_upper** (string texto)

Devuelve **TRUE** si cada caracter del *texto* es una letra mayúscula en la localidad actual.

Ejemplo 1. Un ejemplo de ctype_upper() (usando la localidad predeterminada)

```
<?php
$cadenas = array('AKLWC139', 'LMNSDO', 'akwSKWsm');
foreach ($cadenas as $caso_prueba) {
    if (ctype_upper($caso_prueba)) {
        echo "La cadena $caso_prueba consiste completamente de letras may&uacute;sculas
    } else {
        echo "La cadena $caso_prueba no consiste completamente de letras may&uacute;scu
    }
}
?>
```

Este ejemplo producirá la salida:

```
La cadena AKLWC139 no consiste completamente de signos de letras may&uacute;sculas.
La cadena LMNSDO consiste completamente de signos de letras may&uacute;sculas.
La cadena akwSKWsm no consiste completamente de signos de letras may&uacute;sculas.
```

Vea también **ctype_alpha()**, **ctype_lower()** y **setlocale()**.

ctype_xdigit

(PHP 4 >= 4.0.4, PHP 5)

ctype_xdigit -- Chequear posibles caracteres que representen un dígito hexadecimal

Descripción

bool **ctype_xdigit** (string texto)

Devuelve **TRUE** si cada caracter del *texto* es un 'dígito' hexadecimal, lo que quiere decir un dígito decimal o un caracter del rango *[A-Fa-f]*; **FALSE** de lo contrario.

Ejemplo 1. Un ejemplo de ctype_xdigit()

```
<?php
$cadenas = array('AB10BC99', 'AR1012', 'ab12bc99');
foreach ($cadenas as $caso_prueba) {
    if (ctype_xdigit($caso_prueba)) {
        echo "La cadena $caso_prueba consiste completamente de d&iacute;gitos hexadecim
    } else {
        echo "La cadena $caso_prueba no consiste completamente de d&iacute;gitos hexade
    }
}
?>
```

Este ejemplo producirá la salida:

```
La cadena AB10BC99 consiste completamente de d&iacute;gitos hexadecimales.
La cadena AR1012 no consiste completamente de d&iacute;gitos hexadecimales.
La cadena ab12bc99 consiste completamente de d&iacute;gitos hexadecimales.
```

Vea también **ctype_digit()**.

XVI. Funciones CURL (Client URL Library)

Introducción

PHP soporta libcurl, una biblioteca creada por Danile Stenberg, que permite conexión y comunicación con varios tipos de servidores diferentes con varios tipos de protocolos diferentes. libcurl actualmente soporta los portocolos http, https, ftp, gopher, telnet, dict, file y ldap. libcurl también soporta certificados HTTPS, HTTP POST, HTTP PUT, envío por FTP (esto también puede ser realizado con la extensión ftp de PHP), envío de archivos tipo formulario HTTP, servidores proxy, cookies y autenticación usuario+contraseña.

Estas funciones fueron agregadas en 4.0.2.

Requirimientos

Para poder usar las funciones CURL deberá instalar el paquete CURL. PHP requiere que use CURL 7.0.2-beta o superior. PHP no funcionará con una versión de CURL menor a 7.0.2-beta. Desde la versión 4.2.3 de PHP se necesita, al menos, CURL 7.9.0 o superior.

Instalación

To use PHP's CURL support you must also compile PHP *--with-curl[=DIR]* where DIR is the location of the directory containing the lib and include directories. In the "include" directory there should be a folder named "curl" which should contain the `easy.h` and `curl.h` files. There should be a file named `libcurl.a` located in the "lib" directory. Beginning with PHP 4.3.0 you can configure PHP to use CURL for URL streams *--with-curlwrappers*.

> **Note to Win32 Users:** In order to enable this module on a Windows environment, you must copy `libeay32.dll` and `ssleay32.dll` from the DLL folder of the PHP/Win32 binary package to the SYSTEM folder of your Windows machine. (Ex: `C:\WINNT\SYSTEM32` or `C:\WINDOWS\SYSTEM`)

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

`CURLOPT_DNS_USE_GLOBAL_CACHE` (**integer**)

`CURLOPT_DNS_CACHE_TIMEOUT` (**integer**)

`CURLOPT_FTPSSLAUTH` (**integer**)

> Available since PHP 5.1.0

`CURLOPT_PORT` (**integer**)

`CURLOPT_FILE` (**integer**)

`CURLOPT_INFILE` (**integer**)

`CURLOPT_INFILESIZE` (**integer**)

`CURLOPT_URL` (**integer**)

`CURLOPT_PROXY` (**integer**)

`CURLOPT_VERBOSE` (**integer**)

`CURLOPT_HEADER` (**integer**)

`CURLOPT_HTTPHEADER` (**integer**)

`CURLOPT_NOPROGRESS` (**integer**)

`CURLOPT_NOBODY` (**integer**)

`CURLOPT_FAILONERROR` (**integer**)

CURLOPT_UPLOAD (integer)

CURLOPT_POST (integer)

CURLOPT_FTPLISTONLY (integer)

CURLOPT_FTPAPPEND (integer)

CURLOPT_NETRC (integer)

CURLOPT_FOLLOWLOCATION (integer)

CURLOPT_FTPASCII (integer)

CURLOPT_PUT (integer)

CURLOPT_MUTE (integer)

CURLOPT_USERPWD (integer)

CURLOPT_PROXYUSERPWD (integer)

CURLOPT_RANGE (integer)

CURLOPT_TIMEOUT (integer)

CURLOPT_POSTFIELDS (integer)

CURLOPT_REFERER (integer)

CURLOPT_USERAGENT (integer)

CURLOPT_FTPPORT (integer)

CURLOPT_FTP_USE_EPSV (integer)

CURLOPT_LOW_SPEED_LIMIT (integer)

CURLOPT_LOW_SPEED_TIME (integer)

CURLOPT_RESUME_FROM (integer)

CURLOPT_COOKIE (integer)

CURLOPT_SSLCERT (integer)

CURLOPT_SSLCERTPASSWD (integer)

CURLOPT_WRITEHEADER (integer)

CURLOPT_SSL_VERIFYHOST (integer)

CURLOPT_COOKIEFILE (integer)

CURLOPT_SSLVERSION (integer)

CURLOPT_TIMECONDITION (integer)

CURLOPT_TIMEVALUE (integer)

CURLOPT_CUSTOMREQUEST (integer)

CURLOPT_STDERR (integer)

CURLOPT_TRANSFERTEXT (integer)

CURLOPT_RETURNTRANSFER (integer)

CURLOPT_QUOTE (integer)

CURLOPT_POSTQUOTE (integer)

CURLOPT_INTERFACE (integer)

CURLOPT_KRB4LEVEL (integer)

CURLOPT_HTTPPROXYTUNNEL (integer)

CURLOPT_FILETIME (integer)

CURLOPT_WRITEFUNCTION (integer)

CURLOPT_READFUNCTION (integer)

CURLOPT_PASSWDFUNCTION (integer)

CURLOPT_HEADERFUNCTION (integer)

CURLOPT_MAXREDIRS (integer)

CURLOPT_MAXCONNECTS (integer)

CURLOPT_CLOSEPOLICY (integer)

CURLOPT_FRESH_CONNECT (integer)

CURLOPT_FORBID_REUSE (integer)

CURLOPT_RANDOM_FILE (integer)

CURLOPT_EGDSOCKET (integer)

CURLOPT_CONNECTTIMEOUT (integer)

CURLOPT_SSL_VERIFYPEER (integer)

CURLOPT_CAINFO (integer)

CURLOPT_CAPATH (integer)

CURLOPT_COOKIEJAR (integer)

CURLOPT_SSL_CIPHER_LIST (integer)

CURLOPT_BINARYTRANSFER (integer)

CURLOPT_NOSIGNAL (integer)

CURLOPT_PROXYTYPE (integer)

CURLOPT_BUFFERSIZE (integer)

CURLOPT_HTTPGET (integer)

CURLOPT_HTTP_VERSION (integer)

CURLOPT_SSLKEY (integer)

CURLOPT_SSLKEYTYPE (integer)

CURLOPT_SSLKEYPASSWD (integer)

CURLOPT_SSLENGINE (integer)

CURLOPT_SSLENGINE_DEFAULT (integer)

CURLOPT_SSLCERTTYPE (integer)

CURLOPT_CRLF (integer)

CURLOPT_ENCODING (integer)

CURLOPT_PROXYPORT (integer)

CURLOPT_UNRESTRICTED_AUTH (integer)

CURLOPT_FTP_USE_EPRT (integer)

CURLOPT_HTTP200ALIASES (integer)

CURLOPT_HTTPAUTH (integer)

CURLAUTH_BASIC (integer)

CURLAUTH_DIGEST (integer)

CURLAUTH_GSSNEGOTIATE (integer)

CURLAUTH_NTLM (integer)

CURLAUTH_ANY (integer)

CURLAUTH_ANYSAFE (integer)

CURLOPT_PROXYAUTH (integer)

CURLCLOSEPOLICY_LEAST_RECENTLY_USED (integer)

CURLCLOSEPOLICY_LEAST_TRAFFIC (integer)

CURLCLOSEPOLICY_SLOWEST (integer)

CURLCLOSEPOLICY_CALLBACK (integer)

CURLCLOSEPOLICY_OLDEST (integer)

CURLINFO_EFFECTIVE_URL (integer)

CURLINFO_HTTP_CODE (integer)

CURLINFO_HEADER_SIZE (integer)

CURLINFO_REQUEST_SIZE (integer)

CURLINFO_TOTAL_TIME (integer)

CURLINFO_NAMELOOKUP_TIME (integer)

CURLINFO_CONNECT_TIME (integer)

CURLINFO_PRETRANSFER_TIME (integer)

CURLINFO_SIZE_UPLOAD (integer)

CURLINFO_SIZE_DOWNLOAD (integer)

CURLINFO_SPEED_DOWNLOAD (integer)

CURLINFO_SPEED_UPLOAD (integer)

CURLINFO_FILETIME (integer)

CURLINFO_SSL_VERIFYRESULT (integer)

CURLINFO_CONTENT_LENGTH_DOWNLOAD (integer)

CURLINFO_CONTENT_LENGTH_UPLOAD (integer)

CURLINFO_STARTTRANSFER_TIME (integer)

CURLINFO_CONTENT_TYPE (integer)

CURLINFO_REDIRECT_TIME (integer)

CURLINFO_REDIRECT_COUNT (integer)

CURL_VERSION_IPV6 (integer)

CURL_VERSION_KERBEROS4 (integer)

CURL_VERSION_SSL (integer)

CURL_VERSION_LIBZ (integer)

CURLVERSION_NOW (integer)

CURLE_OK (integer)

CURLE_UNSUPPORTED_PROTOCOL (integer)

CURLE_FAILED_INIT (integer)

CURLE_URL_MALFORMAT (integer)

CURLE_URL_MALFORMAT_USER (integer)

CURLE_COULDNT_RESOLVE_PROXY (integer)

CURLE_COULDNT_RESOLVE_HOST (integer)

CURLE_COULDNT_CONNECT (integer)

CURLE_FTP_WEIRD_SERVER_REPLY (integer)

CURLE_FTP_ACCESS_DENIED (integer)

CURLE_FTP_USER_PASSWORD_INCORRECT (integer)

CURLE_FTP_WEIRD_PASS_REPLY (integer)

CURLE_FTP_WEIRD_USER_REPLY (integer)

CURLE_FTP_WEIRD_PASV_REPLY (integer)

CURLE_FTP_WEIRD_227_FORMAT (integer)

CURLE_FTP_CANT_GET_HOST (integer)

CURLE_FTP_CANT_RECONNECT (integer)

CURLE_FTP_COULDNT_SET_BINARY (integer)

CURLE_PARTIAL_FILE (integer)

CURLE_FTP_COULDNT_RETR_FILE (integer)

CURLE_FTP_WRITE_ERROR (integer)

CURLE_FTP_QUOTE_ERROR (integer)

CURLE_HTTP_NOT_FOUND (integer)

CURLE_WRITE_ERROR (integer)

CURLE_MALFORMAT_USER (integer)

CURLE_FTP_COULDNT_STOR_FILE (integer)

CURLE_READ_ERROR (integer)

CURLE_OUT_OF_MEMORY (integer)

CURLE_OPERATION_TIMEOUTED (integer)

CURLE_FTP_COULDNT_SET_ASCII (integer)

CURLE_FTP_PORT_FAILED (integer)

CURLE_FTP_COULDNT_USE_REST (integer)

CURLE_FTP_COULDNT_GET_SIZE (integer)

CURLE_HTTP_RANGE_ERROR (integer)

CURLE_HTTP_POST_ERROR (integer)

CURLE_SSL_CONNECT_ERROR (integer)

CURLE_FTP_BAD_DOWNLOAD_RESUME (integer)

CURLE_FILE_COULDNT_READ_FILE (integer)

CURLE_LDAP_CANNOT_BIND (integer)

CURLE_LDAP_SEARCH_FAILED (integer)

CURLE_LIBRARY_NOT_FOUND (integer)

CURLE_FUNCTION_NOT_FOUND (integer)

CURLE_ABORTED_BY_CALLBACK (integer)

CURLE_BAD_FUNCTION_ARGUMENT (integer)

CURLE_BAD_CALLING_ORDER (integer)

CURLE_HTTP_PORT_FAILED (integer)

CURLE_BAD_PASSWORD_ENTERED (integer)

CURLE_TOO_MANY_REDIRECTS (integer)

CURLE_UNKNOWN_TELNET_OPTION (integer)

CURLE_TELNET_OPTION_SYNTAX (integer)

CURLE_OBSOLETE (integer)

CURLE_SSL_PEER_CERTIFICATE (integer)

CURLE_GOT_NOTHING (integer)

CURLE_SSL_ENGINE_NOTFOUND (integer)

CURLE_SSL_ENGINE_SETFAILED (integer)

CURLE_SEND_ERROR (integer)

CURLE_RECV_ERROR (integer)

CURLE_SHARE_IN_USE (integer)

CURLE_SSL_CERTPROBLEM (integer)

CURLE_SSL_CIPHER (integer)

CURLE_SSL_CACERT (integer)

CURLE_BAD_CONTENT_ENCODING (integer)

CURLE_LDAP_INVALID_URL (integer)

CURLE_FILESIZE_EXCEEDED (integer)

CURLE_FTP_SSL_FAILED (integer)

CURLFTPAUTH_DEFAULT (integer)

Available since PHP 5.1.0

CURLFTPAUTH_SSL (integer)

Available since PHP 5.1.0

CURLFTPAUTH_TLS (integer)

Available since PHP 5.1.0

CURLPROXY_HTTP (integer)

CURLPROXY_SOCKS5 (integer)

CURL_NETRC_OPTIONAL (integer)

CURL_NETRC_IGNORED (integer)

CURL_NETRC_REQUIRED (integer)

CURL_HTTP_VERSION_NONE (integer)

CURL_HTTP_VERSION_1_0 (integer)

CURL_HTTP_VERSION_1_1 (integer)

CURLM_CALL_MULTI_PERFORM (integer)

CURLM_OK (integer)

CURLM_BAD_HANDLE (integer)

CURLM_BAD_EASY_HANDLE (integer)

CURLM_OUT_OF_MEMORY (integer)

CURLM_INTERNAL_ERROR (integer)

CURLMSG_DONE (integer)

Ejemplos

Una vez que hemos compilado PHP con soporte para CURL, podemos comenzar a usar las funciones CURL. La idea básica tras las funciones CURL es que inicialicemos una sesión CURL usando la función **curl_init()**, luego podemos establecer las opciones para la transferencia a través de la función **curl_setopt()**, y finalmente podemos ejecutar la sesión con la función **curl_exec()** para luego cerrarla con **curl_close()**. Aqui hay un ejemplo que usa funciones CURL para traer el contenido de la página de inicio de example.com y guardarlo en un archivo:

Ejemplo 1. Usando el modulo CURL de PHP para traer la página de inicio de example.com

```
<?php

$ch = curl_init ("http://www.example.com/");
$fp = fopen ("pagina_de_inicio.txt", "w");

curl_setopt ($ch, CURLOPT_FILE, $fp);
curl_setopt ($ch, CURLOPT_HEADER, 0);

curl_exec ($ch);
curl_close ($ch);
fclose ($fp);
?>
```

Tabla de contenidos

curl_close

(PHP 4 >= 4.0.2, PHP 5)

curl_close -- Cierra una sesión CURL

Descripción

void **curl_close** (int ch)

Esta función cierra una sesión CURL y libera todos sus recursos. El recurso CURL, *ch*, también es eliminado.

curl_copy_handle

(PHP 5)

curl_copy_handle -- Copy a cURL handle along with all of it's preferences

Description

resource **curl_copy_handle** (resource ch)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

curl_errno

(PHP 4 >= 4.0.3, PHP 5)

curl_errno -- Devuelve el último número de error

Descripción

int **curl_errno** (resource ch)

Devuelve el número de error de la última operación cURL realizada con el recurso *ch*, o *0* (cero) si no ocurrieron errores.

Vea también **curl_error()**.

curl_error

(PHP 4 >= 4.0.3, PHP 5)

curl_error -- Devuelve una cadena conteniendo el último error para la sesión actual.

Descripción

string **curl_error** (resource ch)

Devuelve un mensaje de error para la última operación cURL realizada con el recurso *ch*, o *''* (una cadena vacía) si no hay error.

Vea también: **curl_errno()**.

curl_exec

(PHP 4 >= 4.0.2, PHP 5)

curl_exec -- Ejecuta una sesión CURL

Descripción

bool **curl_exec** (int ch)

Esta función debe ser llamada luego de inicializar una sesión CURL y fijar todas las opciones para la misma. Su propósito es simplemente el de ejecutar la sesión CURL predefinida (dada por *ch*).

curl_getinfo

(PHP 4 >= 4.0.4, PHP 5)

curl_getinfo -- Obtiene información respecto a una transferencia específica

Descripción

string **curl_getinfo** (resource ch [, int opt])

Devuelve intormación sobre la última transferencia, *opt* puede ser uno de los siguientes:

- "CURLINFO_EFFECTIVE_URL" - La última URL efectiva
- "CURLINFO_HTTP_CODE" - El último código HTTP recibido
- "CURLINFO_FILETIME" - La fecha remota del documento recibido, si el valor devuelto es -1 la fecha del documento es desconocida
- "CURLINFO_TOTAL_TIME" - El tiempo total de transacción en segundos para la última transferencia
- "CURLINFO_NAMELOOKUP_TIME" - El tiempo en segundos transcurrido hasta que la resolución de nombres fue completada
- "CURLINFO_CONNECT_TIME" - El tiempo en segundos que tomó establecer la conexión
- "CURLINFO_PRETRANSFER_TIME" - El tiempo en segundos transcurrido desde el comienzo hasta antes de comenzar la transferencia
- "CURLINFO_STARTTRANSFER_TIME" - El tiempo en segundos transucrrido hasta que el primer byte está listo para ser transferido
- "CURLINFO_REDIRECT_TIME" - El tiempo en segundos de todos los redireccionamientos previos a la transacción final
- "CURLINFO_SIZE_UPLOAD" - Número total de bytes enviados
- "CURLINFO_SIZE_DOWNLOAD" - Número total de bytes recibidos
- "CURLINFO_SPEED_DOWNLOAD" - Velocidad promedio de recepción
- "CURLINFO_SPEED_UPLOAD" - Velocidad promedio de envío
- "CURLINFO_HEADER_SIZE" - Tamaño total de todos los encabezados recibidos
- "CURLINFO_REQUEST_SIZE" - Tamaño total de los requerimientos efectuados, actualmente solo para requerimientos HTTP
- "CURLINFO_SSL_VERIFYRESULT" - Resultado de la verificación de certificado SSL

requerida al indicar CURLOPT_SSL_VERIFYPEER

- "CURLINFO_CONTENT_LENGTH_DOWNLOAD" - El tamaño del contenido recibido, tomado del encabezado Content-Length
- "CURLINFO_CONTENT_LENGTH_UPLOAD" - Tamaño indicado de la información enviada
- "CURLINFO_CONTENT_TYPE" - Tipo de contenido del objeto recibido, NULL indica que el servidor no envío un encabezado Content-Type válido

Si es llamado sin el parámetro opcional *opt* un array asociativo es devuelto con los siguientes elementos, que se corresponden con las opciones del parámetro *opt*:

- "url"
- "content_type"
- "http_encode"
- "header_size"
- "request_size"
- "filetime"
- "ssl_verify_result"
- "redirect_count"
- "total_time"
- "namelookup_time"
- "connect_time"
- "pretransfer_time"
- "size_upload"
- "size_download"
- "speed_download"
- "speed_upload"
- "download_content_length"
- "upload_content_length"
- "starttransfer_time"
- "redirect_time"

curl_init

(PHP 4 >= 4.0.2, PHP 5)

curl_init -- Inicializa una sesión CURL

Descripción

resource **curl_init** ([string url])

curl_init() inicializa una nueva sesión y devuelve un recurso CURL para ser usado con las funciones **curl_setopt()**, **curl_exec()**, y **curl_close()**. Si el parametro opcional *url* es indicado el valor del mismo será asignado a la opción CURLOPT_URL. Esto puede ser asignado manualmente usando la función **curl_setopt()**.

Ejemplo 1. Inicializando una nueva sesión CURL y traer una página web

```
<?php
$ch = curl_init();

curl_setopt ($ch, CURLOPT_URL, "http://www.zend.com/");
curl_setopt ($ch, CURLOPT_HEADER, 0);

curl_exec ($ch);

curl_close ($ch);
?>
```

Vea también: **curl_close()**, **curl_setopt()**

curl_multi_add_handle

(PHP 5)

curl_multi_add_handle -- Add a normal cURL handle to a cURL multi handle

Description

int **curl_multi_add_handle** (resource mh, resource ch)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

See also **curl_multi_init()**, **curl_init()**, and **curl_multi_remove_handle()**.

curl_multi_close

(PHP 5)

curl_multi_close -- Close a set of cURL handles

Description

void **curl_multi_close** (resource mh)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

See also **curl_multi_init()** and **curl_close()**.

curl_multi_exec

(PHP 5)

curl_multi_exec -- Run the sub-connections of the current cURL handle

Description

int **curl_multi_exec** (resource mh, int &still_running)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

See also **curl_multi_init()** and **curl_exec()**.

curl_multi_getcontent

(PHP 5)

curl_multi_getcontent -- Return the content of a cURL handle if CURLOPT_RETURNTRANSFER is set

Description

string **curl_multi_getcontent** (resource ch)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

See also **curl_multi_init()**.

curl_multi_info_read

(PHP 5)

curl_multi_info_read -- Get information about the current transfers

Description

array **curl_multi_info_read** (resource mh)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

See also **curl_multi_init()**.

curl_multi_init

(PHP 5)

curl_multi_init -- Returns a new cURL multi handle

Description

resource **curl_multi_init** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

See also **curl_init()** and **curl_multi_close()**.

curl_multi_remove_handle

(PHP 5)

curl_multi_remove_handle -- Remove a multi handle from a set of cURL handles

Description

int **curl_multi_remove_handle** (resource mh, resource ch)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

See also **curl_multi_init()**, **curl_init()**, and **curl_multi_add_handle()**.

curl_multi_select

(PHP 5)

curl_multi_select -- Get all the sockets associated with the cURL extension, which can then be "selected"

Description

int **curl_multi_select** (resource mh [, float timeout])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

See also **curl_multi_init()**.

curl_setopt

(PHP 4 >= 4.0.2, PHP 5)

curl_setopt -- Asigna un valor a una opción de una sesión CURL

Descripción

bool **curl_setopt** (int ch, string option, mixed value)

La función **curl_setopt()** asigna valores para opciones de una sesión CURL identificada por el parámetro *ch*. El parámetro *option* es la opción a la que se desea asignar el valor indicado en el parámetro *value*.

value debe ser un entero para las siguientes opciones (especificada en el parámetro *option*);

- *CURLOPT_INFILESIZE*: Cuando subimos un archivo a un sitio remoto, esta opción debe ser usada para decirle a PHP cuál será el tamaño del archivo de entrada.
- *CURLOPT_VERBOSE*: Asigne un valor distinto de cero a esta opción si desea que CURL reporte todo lo que acontece.
- *CURLOPT_HEADER*: Asigne un valor distinto de cero a esta opción si desea que el encabezado sea incluido en la salida.
- *CURLOPT_NOPROGRESS*: Asigne un valor distinto de cero a esta opción si no desea que PHP muestre una barra de progreso para las transferencias de CURL.

  Nota: PHP automáticamente asigna un valor distinto de cero a esta opción, esto sólo debe ser cambiado para operaciones de depuración.
- *CURLOPT_NOBODY*: Asigne un valor distinto de cero a esta opción si no desea que el cuerpo sea incluido en la salida.
- *CURLOPT_FAILONERROR*: Asigne un valor distinto de cero a esta opción si desea que PHP falle silenciosamente si el código HTTP devuelto es mayor que 300. El comportamiento por defecto es devolver la página normalmente, ignorando el código.
- *CURLOPT_UPLOAD*: Asigne un valor distinto de cero a esta opción si desea que PHP se prepare para subir un archivo.
- *CURLOPT_POST*: Asigne un valor distinto de cero a esta opción si desea que PHP realize

un pedido HTTP POST regular. Este post normalmente es del tipo application/x-www-from-urlencoded, normalmente utilizado por los formularios HTML.

- *CURLOPT_FTPLISTONLY*: Asigne un valor distinto de cero a esta opción y PHP simplemente listará los nombres de un directorio FTP.

- *CURLOPT_FTPAPPEND*: Asigne un valor distinto de cero a esta opción y PHP escribirá al final del archivo remoto en lugar de sobreescribirlo.

- *CURLOPT_NETRC*: Asigne un valor distinto de cero a esta opción y PHP registrará su archivo ~./netrc para buscar el nombre de usuario y la clave del sitio remoto con el que está estableciendo una conexión.

- *CURLOPT_FOLLOWLOCATION*: Asigne un valor distinto de cero a esta opción para seguir algún encabezado del tipo "Location: " que el servidor envíe como parte de los encabezados HTTP (esto es recursivo, PHP seguirá tantos encabezados "Location: " como le sean enviados).

- *CURLOPT_PUT*: Asigne un valor distinto de cero a esta opción para enviar un archivo utilizando el método HTTP PUT. El archivo a enviar debe ser especificado con las opciones CURLOPT_INFILE y CURLOPT_INFILESIZE.

- *CURLOPT_MUTE*: Asigne un valor distinto de cero a esta opción para que PHP no genere ningúna salida para las funciones CURL.

- *CURLOPT_TIMEOUT*: Asigna un valor que contiene el tiempo máximo de ejecución, en segundos, para las funciones CURL.

- *CURLOPT_LOW_SPEED_LIMIT*: Asigna un valor a la velocidad mínima de transferencia en bytes por segundo. Si la transferencia se mantiene por debajo de ese valor durante la cantidad de segundos indicados en la opción CURLOPT_LOW_SPEED_TIME, será considerada demasiado lenta y abortada por PHP.

- *CURLOPT_LOW_SPEED_TIME*: Asigna un valor a la cantidad de segundos que la transferencia debe permanecer debajo de la velocidad indicada en CURLOPT_LOW_SPEED_LIMIT para que PHP la considere demasiado lenta y aborte.

- *CURLOPT_RESUME_FROM*: Asigna un valor al desplazamiento, en bytes, desde el cual desea que comience la transferencia.

- *CURLOPT_SSLVERSION*: Asigna un valor que contiene la versión de SSL (2 o 3) a usar. Por defecto PHP trata de detectar la versión automáticamente, pero en algúnos casos este valor deberá ser fijado manualmente.

- *CURLOPT_TIMECONDITION*: Asigna un valor que define cómo será tratado CURLOPT_TIMEVALUE. Puede fijar este parametro a TIMECOND_IFMODSINCE o a TIMECOND_ISUNMODSINCE. Esta característica es sólo para HTTP.

- *CURLOPT_TIMEVALUE*: Asigna un valor en segundos al tiempo transcurrido desde el 1 de Enero de 1970. El tiempo será utilizado como sea especificado por la opción CURLOPT_TIMEVALUE, o en su defecto se utilizará TIMECOND_IFMODSINCE.

El parámetro *value* debe ser una cadena para los siguientes valores del parámetro *option*:

- *CURLOPT_URL*: Esta es la URL que se desea traer. Ud. también puede fijar esta opción cuando se inicializa una sesión con la función **curl_init()**.

- *CURLOPT_USERPWD*: Asigna un valor de cadena del tipo [usuario]:[clave], para ser usado por PHP en la conexión.

- *CURLOPT_PROXYUSERPWD*: Asigna un valor de cadena del tipo [usuario]:[clave], para ser usado en la conexión al servidor proxy HTTP.

- *CURLOPT_RANGE*: Asigna un rango en el formato "X-Y", donde X o Y pueden ser dejados de lado. Las transferencias HTTP también soportan varios intervalos, separados por comas como en X-Y,N-M.

- *CURLOPT_POSTFIELDS*: Asigna una cadena que contiene todos los datos a ser enviados en una operación HTTP "POST".

- *CURLOPT_REFERER*: Asigna una cadena que contiene el encabezado "referer" para ser usado en un requerimiento HTTP.

- *CURLOPT_USERAGENT*: Asigna una cadena que contiene el encabezado "user-agent" para ser usado en un requerimiento HTTP.

- *CURLOPT_FTPPORT*: Pasa una cadena que contiene el valor que será usado para obtener la dirección IP a usar en la instrucción "PORT" de ftp. La instrucción "POST" le indica al servidor remoto que se conecte a la dirección IP que le especificamos. La cadena puede ser una dirección IP, un nombre de Host, una interface de red (en UNIX), o simplemente un caracter '-' para usar la dirección IP por defecto del sistema.

- *CURLOPT_COOKIE*: Asigna una cadena con el contenido de la cookie para ser incluida en el encabezado HTTP.

- *CURLOPT_SSLCERT*: Asigna una cadena que contiene el nombre del archivo que guarda el certificado con formato PEM.

- *CURLOPT_SSLCERTPASSWD*: Asigna una cadena que contiene la clave requerida para usar el certificado indicado en la opción CURLOPT_SSLCERT.

- *CURLOPT_COOKIEFILE*: Asigna una cadena que contiene el nombre del archivo que guarda la información de las cookies. Este archivo puede estar en formato Netscape, o simplemente contener encabezados estilo HTTP.

- *CURLOPT_CUSTOMREQUEST*: Asigna una cadena para ser usada en lugar de GET o HEAD cuando se realiza un requerimiento HTTP. Esto es útil para realizar DELETE ú otros requerimientos HTTP más oscuros.

    Nota: No haga esto sin estar seguro de que su servidor soporta el comando especificado.

Las siguientes opciones esperan un descriptor de archivo que es obtenido por medio de la función **fopen()**:

- *CURLOPT_FILE*: El archivo donde debe ser grabada la salida generada por la transferencia. Por defecto es STDOUT.

- *CURLOPT_INFILE*: El archivo desde el que se leen los datos para la transferencia.
- *CURLOPT_WRITEHEADER*: El archivo donde deben ser grabados los encabezados de la salida de la transferencia.
- *CURLOPT_STDERR*: El archivo donde deben ser grabados los errores, en lugar de stderr.

curl_version

(PHP 4 >= 4.0.2, PHP 5)

curl_version -- Devuelve la versión actual de CURL

Descripción

string **curl_version** (void)

La función **curl_version()** devuelve una cadena que contiene la versión actual de CURL.

XVII. Funciones de pago electrónico

Estas funciones solo están disponibles si el intérprete ha sido compilado con *--with-cybercash=[DIR]*. Estas funciones han sido añadidas en PHP4.

Tabla de contenidos

cybercash_base64_decode

(PHP 4 <= 4.2.3)

cybercash_base64_decode --

Descripción

string **cybercash_base64_decode** (string inbuff)

cybercash_base64_encode

(PHP 4 <= 4.2.3)

cybercash_base64_encode -- ???

Descripción

string **cybercash_base64_encode** (string inbuff)

cybercash_decr

(PHP 4 <= 4.2.3)

cybercash_decr -- ???

Descripción

array **cybercash_decr** (string wmk, string sk, string inbuff)

La función devuelve un array asociativo con los elementos "errcode" y, si "errcode" es **FALSE**, "outbuff" (string), "outLth" (long) y "macbuff" (string).

cybercash_encr

(PHP 4 <= 4.2.3)

cybercash_encr -- ???

Descripción

array **cybercash_encr** (string wmk, string sk, string inbuff)

La función devuelve un array asociativo con los elementos "errcode" y, si "errcode" es **FALSE**, "outbuff" (string), "outLth" (long) y "macbuff" (string).

XVIII. Cyrus IMAP administration Functions

Introducción

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

Nota: Esta extensión no está disponible en plataformas Windows

---

Instalación

To enable Cyrus IMAP support and to use these functions you have to compile PHP with the *--with-cyrus* option.

Aviso
La extensión IMAP no puede ser usada junto con las extensiones recode, YAZ ó Cyrus. Esto es debido a que las dos utilizan el mismo símbolo interno

---

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

`CYRUS_CONN_NONSYNCLITERAL` (**integer**)

`CYRUS_CONN_INITIALRESPONSE` (**integer**)

`CYRUS_CALLBACK_NUMBERED` (**integer**)

`CYRUS_CALLBACK_NOLITERAL` (**integer**)

Tabla de contenidos

cyrus_authenticate

(PHP 4 >= 4.1.0)

cyrus_authenticate -- Authenticate against a Cyrus IMAP server

Description

bool **cyrus_authenticate** (resource connection [, string mechlist [, string service [, string user [, int minssf [, int maxssf [, string authname [, string password]]]]]]])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

cyrus_bind

(PHP 4 >= 4.1.0)

cyrus_bind -- Bind callbacks to a Cyrus IMAP connection

Description

bool **cyrus_bind** (resource connection, array callbacks)

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

cyrus_close

(PHP 4 >= 4.1.0)

cyrus_close -- Close connection to a Cyrus IMAP server

Description

bool **cyrus_close** (resource connection)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

cyrus_connect

(PHP 4 >= 4.1.0)

cyrus_connect -- Connect to a Cyrus IMAP server

Description

resource **cyrus_connect** ([string host [, string port [, int flags]]])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

cyrus_query

(PHP 4 >= 4.1.0)

cyrus_query -- Send a query to a Cyrus IMAP server

Description

bool **cyrus_query** (resource connection, string query)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

cyrus_unbind

(PHP 4 >= 4.1.0)

cyrus_unbind -- Unbind ...

Description

bool **cyrus_unbind** (resource connection, string trigger_name)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

XIX. Funciones de Fecha y Hora

Introducción

Estas funciones le permiten obtener la fecha y hora del servidor en donde están siendo ejecutados sus scripts PHP. Puede usar estas funciones para dar formato a las fechas y horas en muchas maneras diferentes.

> **Nota:** Por favor tenga en cuenta que éstas funciones dependen de los parámetros de localidad de su servidor. Asegúrese de tener en cuenta el tiempo de preservación de luz del día y los años bisiestos cuando trabaje con éstas funciones.

Requirimientos

No se necesitan bibliotecas externas para construir esta extensión

---

Instalación

No se necesita ninguna instalación para usar estas funciones, son parte del núcleo de PHP.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Esta extensión no tiene ninguna constante definida.

Tabla de contenidos

checkdate

(PHP 3, PHP 4 , PHP 5)

checkdate -- valida una fecha u hora

Descripción

int **checkdate** (int month, int day, int year)

Devuelve un valor verdadero si la fecha dada es válida; en caso contrario, devuelve un valor falso. Comprueba la validez de la fecha formada por los argumentos. Se considera válida una fecha si:

- el año está entre 0 y 32767, ambos incluidos
- el mes está entre 1 y 12, ambos incluidos
- el día está en el rango permitido para el mes dado. Se tienen en consideración los años bisiestos.

date_sunrise

(PHP 5)

date_sunrise -- Returns time of sunrise for a given day and location

Description

mixed **date_sunrise** (int timestamp [, int format [, float latitude [, float longitude [, float zenith [, float gmt_offset]]]]])

date_sunrise() returns the sunrise time for a given day (specified as a *timestamp*) and location. The *latitude*, *longitude* and *zenith* parameters default to the *date.default_latitude*, *date.default_longitude* and *date.sunrise_zenith* configuration options, respectively.

The *latitude* defaults to North. So, if you want to specify a South value, you must pass a negative value. The same note applies to *longitude*, which defaults to East.

The *gmt_offset* parameter is specified in hours.

Tabla 1. *format* constants

constant	description	example
SUNFUNCS_RET_STRING	returns the sunset time as **string**	16:46
SUNFUNCS_RET_DOUBLE	returns the result as **float**	16.78243132
SUNFUNCS_RET_TIMESTAMP	returns the sunset time as an **integer** (timestamp)	1095034606

Ejemplo 1. date_sunrise() example

```
<?php

/* calculate the sunrise time for Lisbon, Portugal
Latitude: 38.4 North
Longitude: 9 West
Zenith ~= 90
offset: +1 GMT
*/

echo date("D M d Y"). ', sunrise time : ' .date_sunrise(time(), SUNFUNCS_RET_STRING, 38

?>
```

El resultado del ejemplo seria algo similar a:

```
Mon Dec 20 2004, sunrise time : 08:54
```

See also **date_sunset()**.

date_sunset

(PHP 5)

date_sunset -- Returns time of sunset for a given day and location

Description

mixed **date_sunset** (int timestamp [, int format [, float latitude [, float longitude [, float zenith [, float gmt_offset]]]]])

date_sunset() returns the sunset time for a given day (specified as a *timestamp*) and location. The *latitude*, *longitude* and *zenith* parameters default to the *date.default_latitude*, *date.default_longitude* and *date.sunset_zenith* configuration options, respectively.

The *latitude* defaults to North. So, if you want to specify a South value, you must pass a negative value. The same note applies to *longitude*, which defaults to East.

The *gmt_offset* parameter is specified in hours.

Tabla 1. *format* constants

constant	description	example
SUNFUNCS_RET_STRING	returns the sunset time as **string**	16:46
SUNFUNCS_RET_DOUBLE	returns the result as **float**	16.78243132
SUNFUNCS_RET_TIMESTAMP	returns the sunset time as an **integer** (timestamp)	1095034606

Ejemplo 1. date_sunset() example

```
<?php

/* calculate the sunset time for Lisbon, Portugal
Latitude: 38.4 North
Longitude: 9 West
Zenith ~= 90
offset: +1 GMT
*/

echo date("D M d Y"). ', sunset time : ' .date_sunset(time(), SUNFUNCS_RET_STRING, 38.4

?>
```

El resultado del ejemplo seria algo similar a:

```
Mon Dec 20 2004, sunset time : 18:13
```

See also **date_sunrise()**.

date

(PHP 3, PHP 4 , PHP 5)

date -- da formato a la fecha/hora local

Descripción

string **date** (string format [, int timestamp])

Devuelve una cadena formateada de acuerdo con la cadena de formato dada, utilizando el valor de *timestamp* dado o la hora local actual si no hay parámetro.

Se reconocen los siguientes caracteres en la cadena de formato:

- a - "am" o "pm"
- A - "AM" o "PM"
- d - día del mes, dos dígitos con cero a la izquierda; es decir, de "01" a "31"
- D - día de la semana, en texto, con tres letras; por ejemplo, "Fri"
- F - mes, en texto, completo; por ejemplo, "January"

- h - hora, de "01" a "12"
- H - hora, de "00" a "23"
- g - hour, sin ceros, de "1" a "12"
- G - hour, sin ceros; de "0" a "23"
- i - minutos; de "00" a "59"
- j - día del mes sin cero inicial; de "1" a "31"
- l ('L' minúscula) - día de la semana, en texto, completo; por ejemplo, "Friday"
- L - "1" or "0", según si el año es bisiesto o no
- m - mes; de "01" a "12"
- n - mes sin cero inicial; de "1" a "12"
- M - mes, en texto, 3 letras; por ejemplo, "Jan"
- s - segundos; de "00" a "59"
- S - sufijo ordinal en inglés, en texto, 2 caracteres; por ejemplo, "th", "nd"
- t - número de días del mes dado; de "28" a "31"
- U - segundos desde el valor de 'epoch'
- w - día de la semana, en número, de "0" (domingo) a "6" (sábado)
- Y - año, cuatro cifras; por ejemplo, "1999"
- y - año, dos cifras; por ejemplo, "99"
- z - día del año; de "0" a "365"
- Z - diferencia horaria en segundos (de "-43200" a "43200")

Los caracteres no reconocidos se imprimen tal cual. El formato "Z" siempre devuelve "0" en la función **gmdate()()**

Ejemplo 1. Ejemplo de date()

```
print (date("l dS of F Y h:i:s A"));
print ("July 1, 2000 is on a " . date("l", mktime(0,0,0,7,1,2000)));
```

Es posible usar **date()** y **mktime()** juntas para obtener fechas futuras o pasadas.

Ejemplo 2. Ejemplo de date() y mktime()

```
$tomorrow  = mktime(0,0,0,date("m")  ,date("d")+1,date("Y"));
$lastmonth = mktime(0,0,0,date("m")-1,date("d"),  date("Y"));
$nextyear  = mktime(0,0,0,date("m"),  date("d"),  date("Y")+1);
```

Para dar formato a fechas en otros idiomas, se deben usar las funciones **setlocale()** y **strftime()**.

Ver también **gmdate()** y **mktime()**.

getdate

(PHP 3, PHP 4 , PHP 5)

getdate -- obtiene información de fecha y hora

Descripción

array **getdate** (int timestamp)

Devuelve un array asociativo que contiene la información de fecha del valor timestamp como los siguientes elementos:

- "seconds" - segundos
- "minutes" - minutos
- "hours" - horas
- "mday" - día del mes
- "wday" - día de la semana, en número
- "mon" - mes, en número
- "year" - año, en número
- "yday" - día del año, en número; por ejemplo, "299"
- "weekday" - día de la semana, en texto, completo; por ejemplo, "Friday"
- "month" - mes, en texto, completo; por ejemplo, "January"

gettimeofday

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

gettimeofday -- obtiene la hora actual

Descripción

array **gettimeofday** (void)

Es un interfaz para gettimeofday(2). Devuelve un array asociativo que contiene los datos devueltos por esta llamada al sistema.

- "sec" - segundos
- "usec" - microsegundos
- "minuteswest" - minutos al oeste de Greenwich

- "dsttime" - tipo de corrección dst

gmdate

(PHP 3, PHP 4 , PHP 5)

gmdate -- da formato a una fecha/hora GMT/CUT

Descripción

string **gmdate** (string format, int timestamp)

Idéntica a la función **data()** excepto en que la hora devuelta es la de Greenwich (GMT). Por ejemplo, si se utiliza en Finlandia (GMT +0200), la primera línea del ejemplo devuelve "Jan 01 1998 00:00:00", mientras la segunda imprime "Dec 31 1997 22:00:00".

Ejemplo 1. Ejemplo de gmdate()

```
echo date( "M d Y H:i:s",mktime(0,0,0,1,1,1998) );
echo gmdate( "M d Y H:i:s",mktime(0,0,0,1,1,1998) );
```

Ver también **date()**, **mktime()** y **gmmktime()**.

gmmktime

(PHP 3, PHP 4 , PHP 5)

gmmktime -- obtiene el valor timestamp UNIX de una fecha GMT

Descripción

int **gmmktime** (int hour, int minute, int second, int month, int day, int year [, int is_dst])

Idéntica a **mktime()**, excepto en que los parámetros representan una fecha GMT.

gmstrftime

(PHP 3>= 3.0.12, PHP 4 , PHP 5)

gmstrftime -- da formato a una fecha/hora GMT/CUT según las convenciones locales

Descripción

string **gmstrftime** (string format, int timestamp)

Se comporta como **strftime()**, excepto en que la hora devuelta es la de Greenwich (GMT). Por ejemplo, si se utiliza en la zona horaria EST (GMT -0500), la primera línea del ejemplo imprime "Dec 31 1998 20:00:00", mientras la segunda imprime "Jan 01 1999 01:00:00".

Ejemplo 1. Ejemplo de gmstrftime()

```
setlocale ('LC_TIME','en_US');
echo strftime ("%b %d %Y %H:%M:%S",mktime(20,0,0,12,31,98))."\n";
echo gmstrftime ("%b %d %Y %H:%M:%S",mktime(20,0,0,12,31,98))."\n";
```

Ver también **strftime()**.

idate

(PHP 5)

idate -- Format a local time/date as integer

Description

int **idate** (string format [, int timestamp])

Returns a string formatted according to the given format string using the given integer *timestamp* or the current local time if no timestamp is given. In other words, *timestamp* is optional and defaults to the value of **time()**.

Unlike the function **date()**, **idate()** accepts just one char in the *format* parameter.

Tabla 1. The following characters are recognized in the *format* parameter string

format **character**	**Description**
B	Swatch Beat/Internet Time
d	Day of the month
h	Hour (12 hour format)
H	Hour (24 hour format)
i	Minutes
I	returns *1* if DST is activated, *0* otherwise
L	returns *1* for leap year, *0* otherwise
m	Month number
s	Seconds
t	Days in current month
U	Seconds since the Unix Epoch - January 1 1970 00:00:00 GMT - this is the same as **time()**
w	Day of the week (*0* on Sunday)
W	ISO-8601 week number of year, weeks starting on Monday
y	Year (1 or 2 digits - check note below)
Y	Year (4 digits)
z	Day of the year
Z	Timezone offset in seconds

Nota: As **idate()** returns always an **integer** and as they can't start with a "0", **idate()** may return less digits then you would expect. See the example below:

```
<?php
$timestamp = strtotime('1st January 2004'); //1072915200

// this prints the year in a two digit format
// however, as this would start with a "0", it
// only prints "4"
echo idate('y', $timestamp);
?>
```

See also **date()** and **time()**.

localtime

(PHP 4 , PHP 5)

localtime -- Obtener la hora local

Descripción

array **localtime** ([int muestra_de_tiempo [, bool es_asociativo]])

La función **localtime()** devuelve un vector idético al de la estructura devuelta en C por la llamada a la misma función. El primer parámetro que se le pasa a **localtime()** es el timestamp, una representació de una fecha/hora concretas. Si no se proporciona, se utilizará la hora actual. El segundo argumento de **localtime()** es *es_asociativo*. Si está a 0 o no es proporcionado, el vector se devuelve como un vector normal, indizado numéricamente. Si el argumento está a 1, el vector devuelto es un vector asociativo conteniendo los diferentes elementos de la estructura devuelta por C al llamar a la función localtime. Los nombres de las diferentes claves del vector asociativo se encuentran a continuación:

- "tm_sec" - segundos
- "tm_min" - minutos
- "tm_hour" - horas
- "tm_mday" - día del mes
- "tm_mon" - mes del año, empezando en 0 que es Enero
- "tm_year" - Años que hacen desde 1900
- "tm_wday" - Día de la semana
- "tm_yday" - Día del año
- "tm_isdst" - Si el cambio de hora para el ahorro energético tiene efecto o no

microtime

(PHP 3, PHP 4 , PHP 5)

microtime -- devuelve el valor timestamp UNIX actual con microsegundos

Descripción

string **microtime** (void)

Devuelve la cadena "msec sec", donde sec es la hora actual en número de segundos desde el valor Unix Epoch (0:00:00 del 1 de enero de 1970, hora GMT), y msec es la parte de microsegundos. Esta función sólo está disponible en sistemas operativos con admiten la llamada al sistema gettimeofday ().

Ver también **time()**.

mktime

(PHP 3, PHP 4 , PHP 5)

mktime -- obtiene el timestamp UNIX de una fecha

Descripción

int **mktime** (int hour, int minute, int second, int month, int day, int year [, int is_dst])

Advertencia: Véase el extraño orden de los argumentos, que se diferencia del orden de argumentos en una llamada mktime() de UNIX y que no permite eliminar parámetros de derecha a izquierda (ver abajo). Es un error común mezclar estos valores en un script.

Devuelve el valor timestamp Unix correspondiente a los argumentos dados. El timestamp es un entero de tipo long que contiene el número de segundos entre el valor Unix Epoch (1 de enero de 1970) y la hora especificada.

Se pueden eliminar argumentos en orden de derecha a izquierda; en los argumentos omitidos se toma el valor de la fecha y hora locales.

is_dst puede ponerse a 1 si la hora corresponde a horario de verano, 0 si no, o -1 (valor por omisión) si no se sabe.

> **Nota:** *is_dst* se añadió en la versión 3.0.10.

mktime() es útil para realizar cálculos y validaciones con fechas, ya que calcula automáticamente el valor correcto para una entrada fuera de rango. Por ejemplo, cada una de las líneas siguientes produce la cadena "Jan-01-1998".

Ejemplo 1. Ejemplo de mktime()

```
echo date( "M-d-Y", mktime(0,0,0,12,32,1997) );
echo date( "M-d-Y", mktime(0,0,0,13,1,1997) );
echo date( "M-d-Y", mktime(0,0,0,1,1,1998) );
```

El último día de cada mes se puede expresar como el día "0" del mes siguiente, no el día -1. Los dos ejemplos siguientes producen la cadena "The last day in Feb 2000 is: 29".

Ejemplo 2. El último día del próximo mes

```
$lastday=mktime(0,0,0,3,0,2000);
echo strftime("Last day in Feb 2000 is: %d",$lastday);

$lastday=mktime(0,0,0,4,-31,2000);
echo strftime("Last day in Feb 2000 is: %d",$lastday);
```

Ver también **date()** y **time()**.

strftime

(PHP 3, PHP 4 , PHP 5)

strftime -- da formato a la hora o fecha local de acuerdo con las convenciones locales

Descripción

string **strftime** (string format, int timestamp)

Devuelve una cadena formateada según la cadena de formato dada utilizando el valor *timestamp* o la hora local actual. Los nombres del mes y el día de la semana y otras cadenas dependientes del idioma respetan lo establecido con **setlocale()**.

Se reconocen los siguientes especificadores de conversión en la cadena de formato:

- %a - nombre del día de la semana abreviado
- %A - nombre del día de la semana completo
- %b - nombre del mes abreviado
- %B - nombre del mes completo
- %c - representación de fecha y hora preferidas en el idioma actual
- %d - día del mes en número (de 00 a 31)
- %H - hora como un número de 00 a 23
- %I - hora como un número de 01 a 12
- %j - día del año como un número de 001 a 366
- %m - mes como un número de 01 a 12
- %M - minuto en número
- %p - `am' o `pm', según la hora dada, o las cadenas correspondientes en el idioma actual
- %S - segundos en número
- %U - número de la semana en el año, empezando con el primer domingo como el primer día de la primera semana
- %W - número de la semana en el año, empezando con el primer lunes como el primer día de la primera semana
- %w - día de la semana en número (el domingo es el 0)
- %x - representación preferida de la fecha sin la hora

- %X - representación preferida de la hora sin la fecha
- %y - año en número de 00 a 99
- %Y - año en número de cuatro cifras
- %Z - nombre o abreviatura de la zona horaria
- %% - carácter `%'

Ejemplo 1. Ejemplo de strftime()

```
setlocale ("LC_TIME", "C");
print(strftime("%A in Finnish is "));
setlocale ("LC_TIME", "fi_FI");
print(strftime("%A, in French "));
setlocale ("LC_TIME", "fr_CA");
print(strftime("%A and in German "));
setlocale ("LC_TIME", "de_DE");
print(strftime("%A.\n"));
```

Este ejemplo funciona si se tienen los respectivos `locales' instalados en el sistema.

Ver también **setlocale()** y **mktime()**.

strptime

(no version information, might be only in CVS)

strptime -- Parse a time/date generated with **strftime()**

Description

array **strptime** (string timestamp, string format)

strptime() returns an array with the *timestamp* parsed, or **FALSE** on error.

Month and weekday names and other language dependent strings respect the current locale set with **setlocale()** (LC_TIME).

Lista de parámetros

timestamp (**string**)

A timestamp (e.g. returned from **strftime()**)

format (**string**)

The format used in *timestamp* (e.g. the same as used in **strftime()**).

For more information about the format options, read the **strftime()** page.

Valores retornados

Returns an array, or **FALSE** on failure.

Tabla 1. The following parameters are returned in the array

parameters	Description
tm_sec	Seconds after the minute (0-61)
tm_min	Minutes after the hour (0-59)
tm_hour	Hour since midnight (0-23)
tm_mday	Day of the month (1-31)
tm_mon	Months since January (0-11)
tm_year	Years since 1900
tm_wday	Days since Sunday (0-6)
tm_yday	Days since January 1 (0-365)
unparsed	the *timestamp* part which was not recognized using the specified *format*

Ejemplos

Ejemplo 1. strptime() example

```
<?php
$format = '%d/%m/%Y %H:%M:%S';
$strf = strftime($format);

echo "$strf\n";

print_r(strptime($strf, $format));
?>
```

El resultado del ejemplo seria algo similar a:

```
03/10/2004 15:54:19

Array
(
    [tm_sec] => 19
    [tm_min] => 54
    [tm_hour] => 15
    [tm_mday] => 3
    [tm_mon] => 9
    [tm_year] => 104
    [tm_wday] => 0
    [tm_yday] => 276
    [unparsed] =>
)
```

Ver también

strftime()

Nota: Esta función no está implementada en plataformas Windows.

strtotime

(PHP 3>= 3.0.12, PHP 4 , PHP 5)

strtotime -- Procesar cualquier descripción textual de fecha/hora en Inglés convirtiéndola en una timestamp de UNIX.

Descripción

int **strtotime** (string hora [, int ahora])

La función espera que se le pase una cadena conteniendo una fecha en formato Inglés e intentará procesarla y convertirla a una timestamp (muestra de tiempo) de UNIX relativa a la timestamp proporcionada en *ahora*, o la hora actual si no se indica ninguna. Si falla, devolverá *-1*.

Dado que **strtotime()** obra de acuerdo con la sintaxis de fechas de GNU, puede echar un vistazo a la página del manual GNU titulada Date Input Formats (Formatos de entrada de fechas). La sintaxis descrita ahí es válida para el parátro *hora*.

Ejemplo 1. Ejemplos con strtotime()

```
echo strtotime ("now"), "\n";
echo strtotime ("10 September 2000"), "\n";
echo strtotime ("+1 day"), "\n";
echo strtotime ("+1 week"), "\n";
echo strtotime ("+1 week 2 days 4 hours 2 seconds"), "\n";
echo strtotime ("next Thursday"), "\n";
echo strtotime ("last Monday"), "\n";
```

Ejemplo 2. Comprobando si falla

```
$str = 'No v&aacute;lida';
if (($timestamp = strtotime($str)) === -1) {
    echo "La cadena ($str) no es v&aacute;lida.";
} else {
    echo "$str == ". date('l dS of F Y h:i:s A',$timestamp);
}
```

> **Nota:** El rango válido de una timestamp suele ser desde Fri, 13 Dec 1901 20:45:54 GMT (Viernes, 13 de diciembre) a Tue, 19 Jan 2038 03:14:07 GMT (Martes, 19 de enero). (Estas son las fechas que corresponden a los valores mínimo y máximo de un entero con signo de 32 bits.)

time

(PHP 3, PHP 4 , PHP 5)

time -- devuelve el timestamp UNIX actual

Descripción

int **time** (void)

Devuelve la hora actual como número de segundos transcurridos desde las 00:00:00 del 1 de enero de 1970 GMT (Unix Epoch).

Ver también **date()**.

XX. Funciones de la capa de abstraccion de bases de datos (dbm-style)

Estas funciones son la base para el acceso a bases de datos del estilo Berkeley DB.

Este es un nivel de abstraccion general para varias bases de datos. Como tal su funcionalidad esta limitada a un grupo de modernas bases de datos como Sleepycat Software's DB2. (Esta no debe confundirse con IBM DB2 software, la cual es soportada mediante las funciones ODBC.)

El comportamiento de varios aspectos depende de la implementacion de la base de datos. Funciones como **dba_optimize()** y **dba_sync()** cumpliran su funcionalidad con unas bases de datos pero no con otras.

Los siguientes manejadores (handlers) estan soportados:

- dbm es el mas antiguo (original) tipo de base de datos de la familia de Berkeley DB. Se debe evitar su uso, si es posible. Nosotros no soportamos las funciones de compatibilidad de DB2 y gdbm, porque ellas solo son compatibles a nivel de codigo fuente, pero no pueden manejar el formato original dbm.

- ndbm es un tipo mas nuevo y mas flexible que dbm. Todavia tiene la mayoria de las limitaciones de dbm (Por lo tanto es descartado).

- gdbm es el gestor de bases de datos de GNU (database manager).

- db2 es Sleepycat Software's DB2. Es descrito como "un conjunto de herramientas de programacion que proveen acceso de alto nivel a bases de datos en aplicaciones standalone o en el modelo cliente/servidor. "

- cdb es "una rapida, de confianza, sencilla herramienta para la creacion y lectura de bases de datos constantes." Fue creada por el autor de qmail y puede encontrarse en here. Como la base es constante solo se soportan las operaciones de lectura.

Ejemplo 1. Ejemplo de DBA

```
<?php

$id = dba_open("/tmp/test.db", "n", "db2");

if(!$id) {
    echo "dba_open failed\n";
    exit;
}

dba_replace("key", "This is an example!", $id);

if(dba_exists("key", $id)) {
    echo dba_fetch("key", $id);
    dba_delete("key", $id);
}

dba_close($id);
?>
```

DBA es "binary safe" y no tiene ningun limite arbitrario. Hereda todas sus limitaciones de la implementacion de base de datos que tenga.

Todos las bases de datos basadas en ficheros deben proveer un mecanismo para establecer el modo

a la hora de crear nuevas bases de datos, si ello es posible. Habitualmente este modo es pasado como el cuarto argumento en **dba_open()** o en **dba_popen()**.

Se puede acceder a todas las entradas de una base de datos de modo secuencial (lineal) usando las funciones **dba_firstkey()** y **dba_nextkey()**. No se puede cambiar la base de datos mientras se recorre (traversing) por ella.

Ejemplo 2. Recorriendo una base de datos

```
<?php

# ...open database...

$key = dba_firstkey($id);

while($key != false) {
    if(...) { # remember the key to perform some action later
        $handle_later[] = $key;
    }
    $key = dba_nextkey($id);
}

for($i = 0; $i < count($handle_later); $i++)
    dba_delete($handle_later[$i], $id);

?>
```

Tabla de contenidos

dba_close

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

dba_close -- Cerrar uba base de datos

Descripcion

void **dba_close** (int handle)

dba_close() cierra la conexion con una base de datos previamente abierta y libera todos los recursos especificados por *handle*.

handle es un manejador (handle) de la base de datos devuelto por **dba_open()**.

dba_close() no devuelve ningun valor.

Ver tambien: **dba_open()** **dba_popen()**

dba_delete

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

dba_delete -- Borra una entrada especificada por la clave key

Descripcion

bool **dba_delete** (string key, int handle)

dba_delete() borra la entrada especificada por *key* de la base de datos especificada por *handle*.

key es la clave de la entrada que es borrada.

handle es un manejador (handle) de la base de datos devuelto por **dba_open()**.

dba_delete() devuelve **TRUE** o **FALSE**, si la entrada es borrada o no, respectivamente.

Ver tambien: **dba_exists()** **dba_fetch()** **dba_insert()** **dba_replace()**

dba_exists

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

dba_exists -- Comprueba si la clave key existe

Descripcion

bool **dba_exists** (string key, int handle)

dba_exists() comprueba si la clave *key* existe en la base de datos especificada por *handle*.

key es la clave para la que se realiza la comprobacion.

handle es un manejador (handle) de la base de datos devuelto por **dba_open()**.

dba_exists() devuelve **TRUE** o **FALSE**, si la clave es hallada o no, respectivamente.

Ver tambien: **dba_fetch()** **dba_delete()** **dba_insert()** **dba_replace()**

dba_fetch

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

dba_fetch -- Extrae los datos especificados por la clave key

Descripcion

string **dba_fetch** (string key, int handle)

dba_fetch() extrae los datos especificados por la clave *key* de la base de datos determinada por *handle*.

key es la clave de la entrada de los datos que queremos extraer.

handle es un manejador (handle) de la base de datos devuelto por **dba_open()**.

dba_fetch() devuelve la cadena asociada o **FALSE**, si el par key/data es hallado o no, respectivamente.

Ver tambien: **dba_exists()** **dba_delete()** **dba_insert()** **dba_replace()**

dba_firstkey

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

dba_firstkey -- Conseguir la primera clave

Descripcion

string **dba_firstkey** (int handle)

dba_firstkey() devuelve la primera clave de la base de datos especificada por *handle* y resetea el puntero interno de claves. Esto permite una busqueda lineal por toda la base de datos.

handle es un manejador (handle) de la base de datos devuelto por **dba_open()**.

dba_firstkey() devuelve la clave o **FALSE** en funcion de si tiene exito o falla, respectivamente.

Ver tambien: **dba_nextkey()**

dba_handlers

(PHP 4 >= 4.3.0, PHP 5)

dba_handlers -- List all the handlers available

Descripción

array **dba_handlers** ([bool full_info])

dba_handlers() list all the handlers supported by this extension.

Lista de parámetros

full_info

Turns on/off full information display in the result. The default is **FALSE**.

Valores retornados

Returns an array of database handlers. If *full_info* is set to **TRUE**, the array will be associative with the handlers names as keys, and their version information as value. Otherwise, the result will be an indexed array of handlers names.

Nota: When the internal cdb library is used you will see *cdb* and *cdb_make*.

Ejemplos

Ejemplo 1. dba_handlers() Example

```
<?php

echo "Available DBA handlers:\n";
foreach (dba_handlers(true) as $handler_name => $handler_version) {
  // clean the versions
  $handler_version = str_replace('$', '', $handler_version);
  echo " - $handler_name: $handler_version\n";
}

?>
```

El resultado del ejemplo seria algo similar a:

```
Available DBA handlers:
 - cdb: 0.75, Revision: 1.3.2.3
 - cdb_make: 0.75, Revision: 1.2.2.4
 - db2: Sleepycat Software: Berkeley DB 2.7.7: (08/20/99)
 - inifile: 1.0, Revision: 1.6.2.3
 - flatfile: 1.0, Revision: 1.5.2.4
```

dba_insert

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

dba_insert -- Insertar una entrada

Descripcion

bool **dba_insert** (string key, string value, int handle)

dba_insert() inserta la entrada descrita con *key* y *value* dentro de la base de datos especificada por *handle*. Fallara si ya existe una entrada con el mismo parametro *key*.

key es la clave de la entrada a ser insertada.

value es el valor a ser insertado.

handle es un manejador (handle) de la base de datos devuelto por **dba_open()**.

dba_insert() devuelve **TRUE** o **FALSE**, en funcion de si tiene exito o falla, respectivamente.

Ver tambien: **dba_exists()** **dba_delete()** **dba_fetch()** **dba_replace()**

dba_key_split

(PHP 5)

dba_key_split -- Splits a key in string representation into array representation

Descripción

mixed **dba_key_split** (mixed key)

dba_key_split() splits a key (string representation) into an array representation.

Lista de parámetros

key

The key in string representation.

Valores retornados

Returns an array of the form *array(0 => group, 1 => value_name)*. This function will return **FALSE** if *key* is **NULL** or **FALSE**.

Ver también

dba_firstkey()
dba_nextkey()
dba_fetch()

dba_list

(PHP 4 >= 4.3.0, PHP 5)

dba_list -- List all open database files

Descripción

array **dba_list** (void)

dba_list() list all open database files.

Valores retornados

An associative array, in the form *resourceid => filename*.

dba_nextkey

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

dba_nextkey -- Extraer la siguiente clave

Descripcion

string **dba_nextkey** (int handle)

dba_nextkey() devuelve la siguiente clave de la base de datos especificada por *handle* e incrementa el puntero de claves interno.

handle es un manejador (handle) de la base de datos devuelto por **dba_open()**.

dba_nextkey() devuelve la clave o **FALSE** dependiendo de si tiene exito o falla, respectivamente.

Ver tambien: **dba_firstkey()**

dba_open

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

dba_open -- Abrir una base de datos

Descripcion

int **dba_open** (string path, string mode, string handler [, ...])

dba_open() establece una instancia para *path* con *mode* usando *handler*.

path normalmente es el "path" en el sistema de ficheros.

mode es "r" para acceso de lectura, "w" para lectura/escritura de una base de datos ya existente, "c" para lectura/escritura y creacion de una base datos si esta no existe, y "n" para crear, truncar y lectura/escritura.

handler es el nombre de el manejador (handler) que sera usado para el acceso al *path*. Es pasado como un parametro opcional a **dba_open()** y puede usarse en lugar de ella.

dba_open() devuelve un valor positivo de handler o **FALSE**, en el caso de que la apertura de la base de datos se realice o si falla, respectivamente.

Ver tambien: **dba_popen()** **dba_close()**

dba_optimize

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

dba_optimize -- Optimiza la base de datos

Descripcion

bool **dba_optimize** (int handle)

dba_optimize() optimiza la base de datos especificada por *handle*.

handle es un manejador (handle) de la base de datos devuelto por **dba_open()**.

dba_optimize() devuelve **TRUE** o **FALSE**, si la optimizacion tiene exito o falla, respectivamente.

Ver tambien: **dba_sync()**

dba_popen

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

dba_popen -- Apertura persistente de una base de datos

Descripcion

int **dba_popen** (string path, string mode, string handler [, ...])

dba_popen() establece una instancia persistente para *path* con *mode* usando *handler*.

path normalmente es el "path" en el sistema de ficheros.

mode es "r" para acceso de lectura, "w" para lectura/escritura de una base de datos ya existente, "c" para lectura/escritura y creacion de una base datos si esta no existe, y "n" para crear, truncar y lectura/escritura.

handler es el nombre del manejador (handler) que sera usado para el acceso al *path*. Es pasado como un parametro opcional a **dba_popen()** y puede usarse en lugar de ella.

dba_popen() devuelve un valor positivo de handler o **FALSE**, en el caso de que la apertura de la base de datos se realice o si falla, respectivamente.

Ver tambien: **dba_open()** **dba_close()**

dba_replace

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

dba_replace -- Reemplaza o inserta una entrada

Descripcion

bool **dba_replace** (string key, string value, int handle)

dba_replace() reemplaza o inserta la entrada descrita con *key* y *value* dentro de la base de datos especificada por *handle*.

key es la clave de la entrada a insertar.

value es el valor a ser insertado.

handle es un manejador (handle) de la base de datos devuelto por **dba_open()**.

dba_replace() devuelve **TRUE** o **FALSE**, dependiendo de si tiene exito o falla respectivamente.

Ver tambien: **dba_exists()** **dba_delete()** **dba_fetch()** **dba_insert()**

dba_sync

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

dba_sync -- Sincroniza la base de datos

Descripcion

bool **dba_sync** (int handle)

dba_sync() sincroniza la base de datos especificada por *handle*. Esto probablemente realice una escritura fisica en el disco, si es soportado.

handle es un manejador (handle) de la base de datos devuelto por **dba_open()**.

dba_sync() devuelve **TRUE** o **FALSE**, si la sincronizacion tiene exito o falla, respectivamente.

Ver tambien: **dba_optimize()**

XXI. Funciones para dBase

Estas funciones permiten el acceso a datos almacenados en formato dBase (dbf).

No hay soporte para índices o campos Memo. Tampoco hay soporte para bloqueo: si dos procesos concurrentes en el servidor modifican el mismo fichero dBase, probablemente se destruirán los datos.

A diferencia de las bases de datos SQL, las "bases de datos" dBase no pueden cambiar su definición. Una vez creado el fichero, la definición de la base de datos es fija. No hay índices que aceleren la búsqueda u organicen los datos de distinto modo. Los ficheros dBase son simples ficheros secuenciales con registros de longitud fija. Los nuevos registros se añaden al final del fichero y los registros borrados se conservan hasta que se llama a la función **dbase_pack()()**.

Se recomienda no utilizar ficheros dBase como bases de datos, sino elegir cualquier servidor SQL; MySQL o Postgres son opciones habituales con PHP. El soporte para dBase se proporciona para permitir importar y exportar datos a y desde la base de datos web, ya que este formato de ficheros es aceptado habitualmente por las hojas de datos y los organizadores de Windows. La importación y exportación de datos es lo único para lo que sirve el soporte dBase.

Tabla de contenidos

dbase_get_record_with_names -- lee un registro de un fichero dBase como array asociativo
dbase_get_record -- lee un registro de un fichero dBase
dbase_numfields -- cuenta el número de campos en un fichero dBase
dbase_numrecords -- cuenta el número de registros en un fichero dBase
dbase_open -- abre un fichero dBase
dbase_pack -- "empaqueta" un fichero dBase
dbase_replace_record -- reemplaza un registro en un fichero dBase

dbase_add_record

(PHP 3, PHP 4 , PHP 5)

dbase_add_record -- añade un registro a un fichero dBase

Descripción

bool **dbase_add_record** (int dbase_identifier, array record)

Añade los datos de *record* a la base de datos. Si el número de elementos del registro proporcionado no es igual al número de campos de la base de datos, la operación fallará y la función devolverá **`FALSE`**.

dbase_close

(PHP 3, PHP 4 , PHP 5)

dbase_close -- cierra un fichero dBase

Descripción

bool **dbase_close** (int dbase_identifier)

Cierra el fichero asociado con *dbase_identifier*.

dbase_create

(PHP 3, PHP 4 , PHP 5)

dbase_create -- crea una base de datos dBase

Descripción

int **dbase_create** (string filename, array fields)

El parámetro *fields* es un array de arrays, cada uno de los cuales describe el formato de un campo de la base de datos. Cada campo consiste de un nombre, un carácter que indica el tipo de campo, una longitud, y una precisión.

Los tipos de campos disponibles son:

L

Lógico. No tienen longitud ni precisión.

M

Memo. (Sin soporte en PHP.) No tienen longitud ni precisión.

D

Fecha (almacenada como AAAAMMDD). No tienen longitud ni precisión.

N

Número. Tienen longitud y precisión (número de cifras tras el punto decimal).

C

Cadena.

Si la base de datos se crea con éxito, se devuelve un dbase_identifier; en caso contrario, devuelve **FALSE**.

Ejemplo 1. Crear un fichero dBase

```
// "database" name
$dbname = "/tmp/test.dbf";

// database "definition"
$def =
    array(
        array("date",     "D"),
        array("name",     "C",  50),
        array("age",      "N",   3, 0),
        array("email",    "C", 128),
        array("ismember", "L")
    );

// creation
if (!dbase_create($dbname, $def))
    print "<strong>Error!</strong>";
```

dbase_delete_record

(PHP 3, PHP 4 , PHP 5)

dbase_delete_record -- borra un registro del fichero dBase

Descripción

bool **dbase_delete_record** (int dbase_identifier, int record)

Marca el registro *record* para ser borrado del fichero de datos. Para eliminar realmente el registro del fichero, debe llamarse a la función **dbase_pack()**.

dbase_get_header_info

(PHP 5)

dbase_get_header_info -- Gets the header info of a database

Descripción

array **dbase_get_header_info** (int dbase_identifier)

Returns information on the column structure of the given database link identifier.

Lista de parámetros

dbase_identifier

The database link identifier, returned by **dbase_open()** or **dbase_create()**.

Valores retornados

An indexed array with an entry for each column in the database. The array index starts at 0.

Each array element contains an associative array of column information, as described here:

name

The name of the column

type

The human-readable name for the dbase type of the column (i.e. date, boolean, etc.)

length

The number of bytes this column can hold

precision

The number of digits of decimal precision for the column

format

A suggested **printf()** format specifier for the column

offset

The byte offset of the column from the start of the row

If the database header information cannot be read, **FALSE** is returned.

Ejemplos

Ejemplo 1. Showing header information for a dBase database file

```
<?php
// Path to dbase file
$db_path = "/tmp/test.dbf";

// Open dbase file
$dbh = dbase_open($db_path, 0)
  or die("Error! Could not open dbase database file '$db_path'.");

// Get column information
$column_info = dbase_get_header_info($dbh);

// Display information
print_r($column_info);
?>
```

dbase_get_record_with_names

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

dbase_get_record_with_names -- lee un registro de un fichero dBase como array asociativo

Descripción

array **dbase_get_record_with_names** (int dbase_identifier, int record)

Devuelve los datos del registro *record* en un array asociativo. El array incluye también un elemento con índice 'deleted' que vale 1 si el registro ha sido marcado para borrar (ver **dbase_delete_record ()**.

Cada campo se convierte al tipo PHP apropiado. (Las fechas se transforman en cadenas.)

dbase_get_record

(PHP 3, PHP 4 , PHP 5)

dbase_get_record -- lee un registro de un fichero dBase

Descripción

array **dbase_get_record** (int dbase_identifier, int record)

Devuelve los datos del registro *record* en un array. El array se indexa a partir de 0, e incluye un elemento con el índice asociativo 'deleted', que vale 1 si el registro ha sido marcado para borrar (ver **dbase_delete_record()**.

Cada campo se convierte al tipo PHP apropiado. (Las fechas se guardan como cadenas.)

dbase_numfields

(PHP 3, PHP 4 , PHP 5)

dbase_numfields -- cuenta el número de campos en un fichero dBase

Descripción

int **dbase_numfields** (int dbase_identifier)

Devuelve el número de campos (columnas) en el fichero especificado. Los números de campo va de 0 a dbase_numfields($db)-1, mientras los números de registros van de 1 a dbase_numrecords($db).

Ejemplo 1. Uso de dbase_numfields()

```
$rec = dbase_get_record($db, $recno);
$nf  = dbase_numfields($db);
for ($i=0; $i < $nf; $i++) {
    print $rec[$i]."<br>\n";
}
```

dbase_numrecords

(PHP 3, PHP 4 , PHP 5)

dbase_numrecords -- cuenta el número de registros en un fichero dBase

Descripción

int **dbase_numrecords** (int dbase_identifier)

Devuelve el número de registros (filas) en el fichero especificado. Los números de registro van de 1 a dbase_numrecords($db), mientras los números de campo van de 0 a dbase_numfields($db)-1.

dbase_open

(PHP 3, PHP 4 , PHP 5)

dbase_open -- abre un fichero dBase

Descripción

int **dbase_open** (string filename, int flags)

Los "flags" son los que utiliza la llamada al sistema open(). Normalmente, 0 significa sólo lectura, 1 sólo escritura y 2 lectura y escritura.

Devuelve un dbase_identifier del fichero abierto, o **FALSE** si no pudo abrirse el fichero.

dbase_pack

(PHP 3, PHP 4 , PHP 5)

dbase_pack -- "empaqueta" un fichero dBase

Descripción

bool **dbase_pack** (int dbase_identifier)

Empaqueta el fichero especificado, borrando definitivamente todos los registros marcados con la función **dbase_delete_record()**.

dbase_replace_record

(PHP 3>= 3.0.11, PHP 4 , PHP 5)

dbase_replace_record -- reemplaza un registro en un fichero dBase

Descripción

bool **dbase_replace_record** (int dbase_identifier, array record, int dbase_record_number)

Reemplaza los datos asociados con el registro *record_number* con los datos de *record* en el fichero de datos. Si el número de elementos del registro proporcionado no es igual al número de campos de la base de datos, la operación fallará y la función devolverá **`FALSE`**.

dbase_record_number es un entero en el rango de 1 al número de registros en el fichero de datos (devuelto por la función **dbase_numrecords()**).

XXII. Funciones DBM Functions [obsoletas]

Introducción

Estas funciones le permiten almacenar registros almacenados en una base de datos estilo-dbm. Este tipo de base de datos (soportado por la BD de Berkeley, GDBM, y algunas bibliotecas de sistema, asi como en la forma de una biblioteca incorporada de archivos planos) almacena parejas clave/valor (a diferencia de los registros completos soportados por las bases de datos relacionales).

> **Nota:** Sin embargo, el soporte de dbm se considera obsoleto y es recomendable que use las funciones de la capa de abstracción de bases de datos (estilo-dbm) en su lugar.

Requirimientos

Para usar estas funciones es necesario compilar PHP con soporte para una base de datos base. Consulte la lista de bases de datos soportadas.

Instalación

In order to use these functions, you must compile PHP with dbm support by using the *--with-db* option. In addition you must ensure support for an underlying database or you can use some system libraries.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

La función **dbmopen()** devuelve un identificador de base de datos que es usado por las demás funciones dbm.

Constantes predefinidas

Esta extensión no tiene ninguna constante definida.

Ejemplos

Ejemplo 1. Ejemplo de DBM

```
<?php

$dbm = dbmopen("lastseen", "w");
if (dbmexists($dbm, $id_usuario)) {
    $last_seen = dbmfetch($dbm, $id_usuario);
} else {
    dbminsert($dbm, $id_usuario, time());
}
hacer_algo();
dbmreplace($dbm, $id_usuario, time());
dbmclose($dbm);

?>
```

Tabla de contenidos

dblist

(PHP 3, PHP 4)

dblist -- Describe la biblioteca compatible con DBM usada

Descripción

string **dblist** (void)

Ejemplo 1. Obtener la información en la línea de comandos

```
[marcus@zaphod marcus]$ php -r 'echo dblist();'
This is GDBM version 1.8.0, as of May 19, 1999.
```

dbmclose

(PHP 3, PHP 4)

dbmclose -- cierra una base de datos dbm

Descripción

bool **dbmclose** (int identif_dbm)

Desbloquea y cierra la base de datos especificada.

dbmdelete

(PHP 3, PHP 4)

dbmdelete -- borra el valor de una clave de una base de datos dbm

Descripción

bool **dbmdelete** (int identif_dbm, string clave)

Borra el valor para la *clave* en la base de datos.

Devuelve **`FALSE`** si la clave no existía en la base de datos.

dbmexists

(PHP 3, PHP 4)

dbmexists -- dice si existe un valor para una clave dada en la base de datos dbm

Descripción

bool **dbmexists** (int identif_dbm, string clave)

Devuelve **TRUE** si hay un valor asociado con la *clave*.

dbmfetch

(PHP 3, PHP 4)

dbmfetch -- obtiene un valor para una clave desde la base de datos dbm

Descripción

string **dbmfetch** (int identif_dbm, string clave)

Devuelve el valor asociado con la *clave*.

dbmfirstkey

(PHP 3, PHP 4)

dbmfirstkey -- obtiene la primera clave de una base de datos dbm

Descripción

string **dbmfirstkey** (int identif_dbm)

Devuelve la primera clave de la base de datos. Nótese que no se garantiza ningún orden en particular, pues la base de datos se crea utilizando una tabla hash, que no garantiza ordenación alguna.

dbminsert

(PHP 3, PHP 4)

dbminsert -- inserta un valor para una clave en la base de datos dbm

Descripción

int **dbminsert** (int identif_dbm, string clave, string valor)

Añade el valor a la base de datos con la clave especificada.

Devuelve -1 si la base de datos se abrío en modo sólo lectura, 0 si la inserción tuvo éxito y 1 si la clave ya existía (para sustituir el valor, utilice **dbmreplace()**.)

dbmnextkey

(PHP 3, PHP 4)

dbmnextkey -- obtiene la siguiente clave de una base de datos dbm

Descripción

string **dbmnextkey** (int identif_dbm, string clave)

Devuelve la clave que sigue a *clave*. Llamando a **dbmfirstkey()** seguida de llamadas sucesivas a **dbmnextkey()** se pueden visitar todos los pares clave/valor de la base de datos dbm. Por ejemplo:
Ejemplo 1. Visitanco cada par clave/valor en una base de datos dbm.

```
$clave = dbmfirstkey($id_dbm);
while ($clave) {
    echo "$clave = " . dbmfetch($id_dbm, $clave) . "\n";
    $clave = dbmnextkey($id_dbm, $clave);
}
```

dbmopen

(PHP 3, PHP 4)

dbmopen -- Abre una base de datos DBM

Descripción

resource **dbmopen** (string nombre_archivo, string banderas)

El primer argumento es la ruta completa al nombre de archivo de la base DBM a ser abierta, y el segundo es el modo de apertura de archivo, el cual es uno de "r", "n", "c" o "w" para los modos de sólo-lectura, nuevo (implica lectura-escritura, y muy probablemente trunque una base de datos existente con el mismo nombre), crear (implica lectura-escritura, y no truncará una base de datos existente con el mismo nombre) y lectura-escritura respectivamente.

Devuelve un identificador a ser pasado a las demás funciones DBM en caso de éxito, o **FALSE** si ocurre un error.

Si es usado el soporte NDBM, lo que hará NDBM en realidad es crear los archivos `nombre_archivo.dir` y `nombre_archivo.pag`. GDBM usa sólo un archivo, asi como lo hace el soporte interno de archivos planos, y la BD Berkeley crea un archivo `nombre_archivo.db`. Note que PHP realiza su propio bloqueo de archivos adicionalmente a cualquier bloqueo de archivos que pueda ser efectuado por la biblioteca DBM misma. PHP no elimina los archivos `.lck` que crea. Estos archivos son usados simplemente como inodes fijos sobre los que se realiza el bloqueo de archivos. Para más información sobre los archivos DBM, consulte sus páginas man Unix, u obtenga GNU GDBM.

> **Nota:** Cuando safe-mode (modo-seguro) está activado, PHP comprueba si los archivos o directorios que va a utilizar tienen la misma UID que el script que está siendo ejecutado.

dbmreplace

(PHP 3, PHP 4)

dbmreplace -- sustituye el valor de una clave en la base de datos dbm

Descripción

bool **dbmreplace** (int identif_dbm, string clave, string valor)

Sustituye el valor para la clave especificada de la base de datos.

También añadirá la clave a la base de datos si no existía antes.

XXIII. DB++ Functions

Aviso
Esta extensión es *EXPERIMENTAL*. Esto significa que el comportamiento de esta extensión, los nombre de sus funciones y en definitiva TODO lo documentado sobre esta extensión, puede cambiar en una futura versión de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Introducción

db++, made by the German company Concept asa, is a relational database system with high performance and low memory and disk usage in mind. While providing SQL as an additional language interface, it is not really a SQL database in the first place but provides its own AQL query language which is much more influenced by the relational algebra then SQL is.

Concept asa always had an interest in supporting open source languages, db++ has had Perl and Tcl call interfaces for years now and uses Tcl as its internal stored procedure language.

Requirimientos

This extension relies on external client libraries so you have to have a db++ client installed on the system you want to use this extension on.

Concept asa provides db++ Demo versions and documentation for Linux, some other Unix versions. There is also a Windows version of db++, but this extension doesn't support it (yet).

Instalación

In order to build this extension yourself you need the db++ client libraries and header files to be installed on your system (these are included in the db++ installation archives by default). You have to run **configure** with option *--with-dbplus* to build this extension.

configure looks for the client libraries and header files under the default paths `/usr/dbplus`, `/usr/local/dbplus` and `/opt/dblus`. If you have installed db++ in a different place you have add the installation path to the **configure** option like this: *--with-dbplus=/your/installation/path*.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

dbplus_relation

Most db++ functions operate on or return *dbplus_relation* resources. A *dbplus_relation* is a handle to a stored relation or a relation generated as the result of a query.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

---

db++ error codes

Tabla 1. DB++ Error Codes

PHP Constant	db++ constant	meaning
DBPLUS_ERR_NOERR (**integer**)	ERR_NOERR	Null error condition
DBPLUS_ERR_DUPLICATE (**integer**)	ERR_DUPLICATE	Tried to insert a duplicate tuple
DBPLUS_ERR_EOSCAN (**integer**)	ERR_EOSCAN	End of scan from rget()
DBPLUS_ERR_EMPTY (**integer**)	ERR_EMPTY	Relation is empty (server)
DBPLUS_ERR_CLOSE (**integer**)	ERR_CLOSE	The server can't close
DBPLUS_ERR_WLOCKED (**integer**)	ERR_WLOCKED	The record is write locked
DBPLUS_ERR_LOCKED (**integer**)	ERR_LOCKED	Relation was already locked
DBPLUS_ERR_NOLOCK (**integer**)	ERR_NOLOCK	Relation cannot be locked

PHP Constant	db++ constant	meaning
DBPLUS_ERR_READ (integer)	ERR_READ	Read error on relation
DBPLUS_ERR_WRITE (integer)	ERR_WRITE	Write error on relation
DBPLUS_ERR_CREATE (integer)	ERR_CREATE	Create() system call failed
DBPLUS_ERR_LSEEK (integer)	ERR_LSEEK	Lseek() system call failed
DBPLUS_ERR_LENGTH (integer)	ERR_LENGTH	Tuple exceeds maximum length
DBPLUS_ERR_OPEN (integer)	ERR_OPEN	Open() system call failed
DBPLUS_ERR_WOPEN (integer)	ERR_WOPEN	Relation already opened for writing
DBPLUS_ERR_MAGIC (integer)	ERR_MAGIC	File is not a relation
DBPLUS_ERR_VERSION (integer)	ERR_VERSION	File is a very old relation
DBPLUS_ERR_PGSIZE (integer)	ERR_PGSIZE	Relation uses a different page size
DBPLUS_ERR_CRC (integer)	ERR_CRC	Invalid crc in the superpage
DBPLUS_ERR_PIPE (integer)	ERR_PIPE	Piped relation requires lseek()
DBPLUS_ERR_NIDX (integer)	ERR_NIDX	Too many secondary indices
DBPLUS_ERR_MALLOC (integer)	ERR_MALLOC	Malloc() call failed
DBPLUS_ERR_NUSERS (integer)	ERR_NUSERS	Error use of max users
DBPLUS_ERR_PREEXIT (integer)	ERR_PREEXIT	Caused by invalid usage
DBPLUS_ERR_ONTRAP (integer)	ERR_ONTRAP	Caused by a signal
DBPLUS_ERR_PREPROC (integer)	ERR_PREPROC	Error in the preprocessor
DBPLUS_ERR_DBPARSE (integer)	ERR_DBPARSE	Error in the parser
DBPLUS_ERR_DBRUNERR (integer)	ERR_DBRUNERR	Run error in db
DBPLUS_ERR_DBPREEXIT (integer)	ERR_DBPREEXIT	Exit condition caused by prexit() * procedure
DBPLUS_ERR_WAIT (integer)	ERR_WAIT	Wait a little (Simple only)

PHP Constant	db++ constant	meaning
DBPLUS_ERR_CORRUPT_TUPLE (integer)	ERR_CORRUPT_TUPLE	A client sent a corrupt tuple
DBPLUS_ERR_WARNING0 (integer)	ERR_WARNING0	The Simple routines encountered a non fatal error which was corrected
DBPLUS_ERR_PANIC (integer)	ERR_PANIC	The server should not really die but after a disaster send ERR_PANIC to all its clients
DBPLUS_ERR_FIFO (integer)	ERR_FIFO	Can't create a fifo
DBPLUS_ERR_PERM (integer)	ERR_PERM	Permission denied
DBPLUS_ERR_TCL (integer)	ERR_TCL	TCL_error
DBPLUS_ERR_RESTRICTED (integer)	ERR_RESTRICTED	Only two users
DBPLUS_ERR_USER (integer)	ERR_USER	An error in the use of the library by an application programmer
DBPLUS_ERR_UNKNOWN (integer)	ERR_UNKNOWN	

Tabla de contenidos

dbplus_add

(4.1.0 - 4.2.3 only)

dbplus_add -- Add a tuple to a relation

Description

int **dbplus_add** (resource relation, array tuple)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

This function will add a tuple to a relation. The *tuple* data is an array of attribute/value pairs to be inserted into the given *relation*. After successful execution the *tuple* array will contain the complete data of the newly created tuple, including all implicitly set domain fields like sequences.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See **dbplus_errcode()** or the introduction to this chapter for more information on db++ error codes.

dbplus_aql

(4.1.0 - 4.2.3 only)

dbplus_aql -- Perform AQL query

Description

resource **dbplus_aql** (string query [, string server [, string dbpath]])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

dbplus_aql() will execute an AQL *query* on the given *server* and *dbpath*.

On success it will return a relation handle. The result data may be fetched from this relation by calling **dbplus_next()** and **dbplus_current()**. Other relation access functions will not work on a result relation.

Further information on the AQL A... Query Language is provided in the original db++ manual.

dbplus_chdir

(4.1.0 - 4.2.3 only)

dbplus_chdir -- Get/Set database virtual current directory

Description

string **dbplus_chdir** ([string newdir])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

dbplus_chdir() will change the virtual current directory where relation files will be looked for by **dbplus_open()**. **dbplus_chdir()** will return the absolute path of the current directory. Calling **dbplus_chdir()** without giving any *newdir* may be used to query the current working directory.

dbplus_close

(4.1.0 - 4.2.3 only)

dbplus_close -- Close a relation

Description

int **dbplus_close** (resource relation)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Calling **dbplus_close()** will close a relation previously opened by **dbplus_open()**.

dbplus_curr

(4.1.0 - 4.2.3 only)

dbplus_curr -- Get current tuple from relation

Description

int **dbplus_curr** (resource relation, array &tuple)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

dbplus_curr() will read the data for the current tuple for the given *relation* and will pass it back as an associative array in *tuple*.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See **dbplus_errcode()** or the introduction to this chapter for more information on db++ error codes.

See also **dbplus_first()**, **dbplus_prev()**, **dbplus_next()**, and **dbplus_last()**.

dbplus_errcode

(4.1.0 - 4.2.3 only)

dbplus_errcode -- Get error string for given errorcode or last error

Description

string **dbplus_errcode** ([int errno])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

dbplus_errcode() returns a cleartext error string for the error code passed as *errno* of for the result code of the last db++ operation if no parameter is given.

dbplus_errno

(4.1.0 - 4.2.3 only)

dbplus_errno -- Get error code for last operation

Description

int **dbplus_errno** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

dbplus_errno() will return the error code returned by the last db++ operation.

See also **dbplus_errcode()**.

dbplus_find

(4.1.0 - 4.2.3 only)

dbplus_find -- Set a constraint on a relation

Description

int **dbplus_find** (resource relation, array constraints, mixed tuple)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

dbplus_find() will place a constraint on the given relation. Further calls to functions like **dbplus_curr()** or **dbplus_next()** will only return tuples matching the given constraints.

Constraints are triplets of strings containing of a domain name, a comparison operator and a comparison value. The *constraints* parameter array may consist of a collection of string arrays, each of which contains a domain, an operator and a value, or of a single string array containing a multiple of three elements.

The comparison operator may be one of the following strings: '==', '>', '>=', '<', '<=', '!=', '~' for a regular expression match and 'BAND' or 'BOR' for bitwise operations.

See also **dbplus_unselect()**.

dbplus_first

(4.1.0 - 4.2.3 only)

dbplus_first -- Get first tuple from relation

Description

int **dbplus_first** (resource relation, array &tuple)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

dbplus_curr() will read the data for the first tuple for the given *relation*, make it the current tuple and pass it back as an associative array in *tuple*.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See **dbplus_errcode()** or the introduction to this chapter for more information on db++ error codes.

See also **dbplus_curr()**, **dbplus_prev()**, **dbplus_next()**, and **dbplus_last()**.

dbplus_flush

(4.1.0 - 4.2.3 only)

dbplus_flush -- Flush all changes made on a relation

Description

int **dbplus_flush** (resource relation)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

dbplus_flush() will write all changes applied to *relation* since the last flush to disk.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See **dbplus_errcode()** or the introduction to this chapter for more information on db++ error codes.

dbplus_freealllocks

(4.1.0 - 4.2.3 only)

dbplus_freealllocks -- Free all locks held by this client

Description

int **dbplus_freealllocks** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

dbplus_freealllocks() will free all tuple locks held by this client.

See also **dbplus_getlock()**, **dbplus_freelock()**, and **dbplus_freerlocks()**.

dbplus_freelock

(4.1.0 - 4.2.3 only)

dbplus_freelock -- Release write lock on tuple

Description

int **dbplus_freelock** (resource relation, string tname)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

dbplus_freelock() will release a write lock on the given *tuple* previously obtained by **dbplus_getlock()**.

See also **dbplus_getlock()**, **dbplus_freerlocks()**, and **dbplus_freealllocks()**.

dbplus_freerlocks

(4.1.0 - 4.2.3 only)

dbplus_freerlocks -- Free all tuple locks on given relation

Description

int **dbplus_freerlocks** (resource relation)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

dbplus_freerlocks() will free all tuple locks held on the given *relation*.

See also **dbplus_getlock()**, **dbplus_freelock()**, and **dbplus_freealllocks()**.

dbplus_getlock

(4.1.0 - 4.2.3 only)

dbplus_getlock -- Get a write lock on a tuple

Description

int **dbplus_getlock** (resource relation, string tname)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

dbplus_getlock() will request a write lock on the specified *tuple*. It will return zero on success or a non-zero error code, especially DBPLUS_ERR_WLOCKED, on failure.

See also **dbplus_freelock()**, **dbplus_freerlocks()**, and **dbplus_freealllocks()**.

dbplus_getunique

(4.1.0 - 4.2.3 only)

dbplus_getunique -- Get an id number unique to a relation

Description

int **dbplus_getunique** (resource relation, int uniqueid)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

dbplus_getunique() will obtain a number guaranteed to be unique for the given *relation* and will pass it back in the variable given as *uniqueid*.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See **dbplus_errcode()** or the introduction to this chapter for more information on db++ error codes.

dbplus_info

(4.1.0 - 4.2.3 only)

dbplus_info -- Get information about a relation

Description

int **dbplus_info** (resource relation, string key, array &result)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

dbplus_last

(4.1.0 - 4.2.3 only)

dbplus_last -- Get last tuple from relation

Description

int **dbplus_last** (resource relation, array &tuple)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

dbplus_curr() will read the data for the last tuple for the given *relation*, make it the current tuple and pass it back as an associative array in *tuple*.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See **dbplus_errcode()** or the introduction to this chapter for more information on db++ error codes.

See also **dbplus_first()**, **dbplus_curr()**, **dbplus_prev()**, and **dbplus_next()**.

dbplus_lockrel

(no version information, might be only in CVS)

dbplus_lockrel -- Request write lock on relation

Description

int **dbplus_lockrel** (resource relation)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

dbplus_lockrel() will request a write lock on the given relation. Other clients may still query the relation, but can't alter it while it is locked.

dbplus_next

(4.1.0 - 4.2.3 only)

dbplus_next -- Get next tuple from relation

Description

int **dbplus_next** (resource relation, array &tuple)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

dbplus_curr() will read the data for the next tuple for the given *relation*, will make it the current tuple and will pass it back as an associative array in *tuple*.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See **dbplus_errcode()** or the introduction to this chapter for more information on db++ error codes.

See also **dbplus_first()**, **dbplus_curr()**, **dbplus_prev()**, and **dbplus_last()**.

dbplus_open

(4.1.0 - 4.2.3 only)

dbplus_open -- Open relation file

Description

resource **dbplus_open** (string name)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

The relation file *name* will be opened. *name* can be either a file name or a relative or absolute path

name. This will be mapped in any case to an absolute relation file path on a specific host machine and server.

On success a relation file resource (cursor) is returned which must be used in any subsequent commands referencing the relation. Failure leads to a zero return value, the actual error code may be asked for by calling **dbplus_errno()**.

dbplus_prev

(4.1.0 - 4.2.3 only)

dbplus_prev -- Get previous tuple from relation

Description

int **dbplus_prev** (resource relation, array &tuple)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

dbplus_curr() will read the data for the previous tuple for the given *relation*, will make it the current tuple and will pass it back as an associative array in *tuple*.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See **dbplus_errcode()** or the introduction to this chapter for more information on db++ error codes.

See also **dbplus_first()**, **dbplus_curr()**, **dbplus_next()**, and **dbplus_last()**.

dbplus_rchperm

(4.1.0 - 4.2.3 only)

dbplus_rchperm -- Change relation permissions

Description

int **dbplus_rchperm** (resource relation, int mask, string user, string group)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

dbplus_rchperm() will change access permissions as specified by *mask*, *user* and *group*. The values for these are operating system specific.

dbplus_rcreate

(4.1.0 - 4.2.3 only)

dbplus_rcreate -- Creates a new DB++ relation

Description

resource **dbplus_rcreate** (string name, mixed domlist [, bool overwrite])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

dbplus_rcreate() will create a new relation named *name*. An existing relation by the same name will only be overwritten if the relation is currently not in use and *overwrite* is set to TRUE.

domlist should contain the domain specification for the new relation within an array of domain description strings. (**dbplus_rcreate()** will also accept a string with space delimited domain description strings, but it is recommended to use an array). A domain description string consists of a domain name unique to this relation, a slash and a type specification character. See the db++ documentation, especially the dbcreate(1) manpage, for a description of available type specifiers and their meanings.

dbplus_rcrtexact

(4.1.0 - 4.2.3 only)

dbplus_rcrtexact -- Creates an exact but empty copy of a relation including indices

Description

resource **dbplus_rcrtexact** (string name, resource relation [, bool overwrite])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

dbplus_rcrtexact() will create an exact but empty copy of the given *relation* under a new *name*. An existing relation by the same *name* will only be overwritten if *overwrite* is **`TRUE`** and no other process is currently using the relation.

dbplus_rcrtlike

(4.1.0 - 4.2.3 only)

dbplus_rcrtlike -- Creates an empty copy of a relation with default indices

Description

resource **dbplus_rcrtlike** (string name, resource relation [, int overwrite])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

dbplus_rcrtexact() will create an empty copy of the given *relation* under a new *name*, but with default indices. An existing relation by the same *name* will only be overwritten if *overwrite* is **TRUE** and no other process is currently using the relation.

dbplus_resolve

(4.1.0 - 4.2.3 only)

dbplus_resolve -- Resolve host information for relation

Description

int **dbplus_resolve** (string relation_name)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

dbplus_resolve() will try to resolve the given *relation_name* and find out internal server id, real hostname and the database path on this host. The function will return an array containing these values under the keys 'sid', 'host' and 'host_path' or **FALSE** on error.

See also **dbplus_tcl()**.

dbplus_restorepos

(4.1.0 - 4.2.3 only)

dbplus_restorepos -- Restore position

Description

int **dbplus_restorepos** (resource relation, array tuple)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

dbplus_rkeys

(4.1.0 - 4.2.3 only)

dbplus_rkeys -- Specify new primary key for a relation

Description

resource **dbplus_rkeys** (resource relation, mixed domlist)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

dbplus_rkeys() will replace the current primary key for *relation* with the combination of domains specified by *domlist*.

domlist may be passed as a single domain name string or as an array of domain names.

dbplus_ropen

(4.1.0 - 4.2.3 only)

dbplus_ropen -- Open relation file local

Description

resource **dbplus_ropen** (string name)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

dbplus_ropen() will open the relation *file* locally for quick access without any client/server overhead. Access is read only and only **dbplus_current()** and **dbplus_next()** may be applied to the returned relation.

dbplus_rquery

(4.1.0 - 4.2.3 only)

dbplus_rquery -- Perform local (raw) AQL query

Description

int **dbplus_rquery** (string query [, string dbpath])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

dbplus_rquery() performs a local (raw) AQL query using an AQL interpreter embedded into the db++ client library. **dbplus_rquery()** is faster than **dbplus_aql()** but will work on local data only.

dbplus_rrename

(4.1.0 - 4.2.3 only)

dbplus_rrename -- Rename a relation

Description

int **dbplus_rrename** (resource relation, string name)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

dbplus_rrename() will change the name of *relation* to *name*.

dbplus_rsecindex

(4.1.0 - 4.2.3 only)

dbplus_rsecindex -- Create a new secondary index for a relation

Description

resource **dbplus_rsecindex** (resource relation, mixed domlist, int type)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

dbplus_rsecindex() will create a new secondary index for *relation* with consists of the domains specified by *domlist* and is of type *type*

domlist may be passed as a single domain name string or as an array of domain names.

dbplus_runlink

(4.1.0 - 4.2.3 only)

dbplus_runlink -- Remove relation from filesystem

Description

int **dbplus_runlink** (resource relation)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

dbplus_unlink() will close and remove the *relation*.

dbplus_rzap

(4.1.0 - 4.2.3 only)

dbplus_rzap -- Remove all tuples from relation

Description

int **dbplus_rzap** (resource relation)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

dbplus_rzap() will remove all tuples from *relation*.

dbplus_savepos

(4.1.0 - 4.2.3 only)

dbplus_savepos -- Save position

Description

int **dbplus_savepos** (resource relation)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

dbplus_setindex

(4.1.0 - 4.2.3 only)

dbplus_setindex -- Set index

Description

int **dbplus_setindex** (resource relation, string idx_name)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

dbplus_setindexbynumber

(4.1.0 - 4.2.3 only)

dbplus_setindexbynumber -- Set index by number

Description

int **dbplus_setindexbynumber** (resource relation, int idx_number)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

dbplus_sql

(4.1.0 - 4.2.3 only)

dbplus_sql -- Perform SQL query

Description

resource **dbplus_sql** (string query [, string server [, string dbpath]])

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

dbplus_tcl

(4.1.0 - 4.2.3 only)

dbplus_tcl -- Execute TCL code on server side

Description

int **dbplus_tcl** (int sid, string script)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

A db++ server will prepare a TCL interpreter for each client connection. This interpreter will enable the server to execute TCL code provided by the client as a sort of stored procedures to improve the performance of database operations by avoiding client/server data transfers and context switches.

dbplus_tcl() needs to pass the client connection id the TCL *script* code should be executed by.

dbplus_resolve() will provide this connection id. The function will return whatever the TCL code returns or a TCL error message if the TCL code fails.

See also **dbplus_resolve()**.

dbplus_tremove

(4.1.0 - 4.2.3 only)

dbplus_tremove -- Remove tuple and return new current tuple

Description

int **dbplus_tremove** (resource relation, array tuple [, array &current])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

dbplus_tremove() removes *tuple* from *relation* if it perfectly matches a tuple within the relation. *current*, if given, will contain the data of the new current tuple after calling **dbplus_tremove()**.

dbplus_undo

(4.1.0 - 4.2.3 only)

dbplus_undo -- Undo

Description

int **dbplus_undo** (resource relation)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.
Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

dbplus_undoprepare

(4.1.0 - 4.2.3 only)

dbplus_undoprepare -- Prepare undo

Description

int **dbplus_undoprepare** (resource relation)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

dbplus_unlockrel

(4.1.0 - 4.2.3 only)

dbplus_unlockrel -- Give up write lock on relation

Description

int **dbplus_unlockrel** (resource relation)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

dbplus_unlockrel() will release a write lock previously obtained by **dbplus_lockrel()**.

dbplus_unselect

(4.1.0 - 4.2.3 only)

dbplus_unselect -- Remove a constraint from relation

Description

int **dbplus_unselect** (resource relation)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Calling **dbplus_unselect()** will remove a constraint previously set by **dbplus_find()** on *relation*.

dbplus_update

(4.1.0 - 4.2.3 only)

dbplus_update -- Update specified tuple in relation

Description

int **dbplus_update** (resource relation, array old, array new)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

dbplus_update() replaces the tuple given by *old* with the data from *new* if and only if *old* completely matches a tuple within *relation*.

dbplus_xlockrel

(4.1.0 - 4.2.3 only)

dbplus_xlockrel -- Request exclusive lock on relation

Description

int **dbplus_xlockrel** (resource relation)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

dbplus_xlockrel() will request an exclusive lock on *relation* preventing even read access from other clients.

See also **dbplus_xunlockrel()**.

dbplus_xunlockrel

(4.1.0 - 4.2.3 only)

dbplus_xunlockrel -- Free exclusive lock on relation

Description

int **dbplus_xunlockrel** (resource relation)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

dbplus_xunlockrel() will release an exclusive lock on *relation* previously obtained by **dbplus_xlockrel()**.

XXIV. dbx Functions

Introducción

The dbx module is a database abstraction layer (db 'X', where 'X' is a supported database). The dbx functions allow you to access all supported databases using a single calling convention. The dbx-functions themselves do not interface directly to the databases, but interface to the modules that are used to support these databases.

Requirimientos

To be able to use a database with the dbx-module, the module must be either linked or loaded into PHP, and the database module must be supported by the dbx-module. Currently, the following databases are supported, but others will follow:

- FrontBase (available from PHP 4.1.0).
- Microsoft SQL Server
- MySQL
- ODBC
- PostgreSQL
- Sybase-CT (available from PHP 4.2.0).
- Oracle (oci8) (available from PHP 4.3.0).
- SQLite (PHP 5).

Documentation for adding additional database support to dbx can be found at http://www.guidance.nl/php/dbx/doc/.

Instalación

In order to have these functions available, you must compile PHP with dbx support by using the *--enable-dbx* option and all options for the databases that will be used, e.g. for MySQL you must also specify *--with-mysql=[DIR]*. To get other supported databases to work with the dbx-module refer to their specific documentation.

Configuración en tiempo de ejecución

El comportamiento de estas funciones está afectado por los valores definidos en `php.ini`.

Tabla 1. DBX Configuration Options

Name	Default	Changeable
dbx.colnames_case	"unchanged"	PHP_INI_SYSTEM

For further details and definitions of the PHP_INI_* constants, see the Apéndice H.

> **Nota:** This ini-option is available available from PHP 4.3.0.

A continuación se presenta una corta explicación de las directivas de configuración.

dbx.colnames_case **string**

> Columns names can be returned "unchanged" or converted to "uppercase" or "lowercase". This directive can be overridden with a flag to **dbx_query()**.

Tipos de recursos

There are two resource types used in the dbx module. The first one is the link-**object** for a database connection, the second a result-**object** which holds the result of a query.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

`DBX_MYSQL` (**integer**)

`DBX_ODBC` (**integer**)

`DBX_PGSQL` (**integer**)

`DBX_MSSQL` (**integer**)

`DBX_FBSQL` (**integer**)

`DBX_OCI8` (**integer**) (available from PHP 4.3.0)

`DBX_SYBASECT` (**integer**)

`DBX_SQLITE` (**integer**) (PHP 5)

DBX_PERSISTENT (integer)

DBX_RESULT_INFO (integer)

DBX_RESULT_INDEX (integer)

DBX_RESULT_ASSOC (integer)

DBX_RESULT_UNBUFFERED (integer) (PHP 5)

DBX_COLNAMES_UNCHANGED (integer) (available from PHP 4.3.0)

DBX_COLNAMES_UPPERCASE (integer) (available from PHP 4.3.0)

DBX_COLNAMES_LOWERCASE (integer) (available from PHP 4.3.0)

DBX_CMP_NATIVE (integer)

DBX_CMP_TEXT (integer)

DBX_CMP_NUMBER (integer)

DBX_CMP_ASC (integer)

DBX_CMP_DESC (integer)

Tabla de contenidos

dbx_close

(PHP 4 >= 4.0.6, PHP 5)

dbx_close -- Close an open connection/database

Description

bool **dbx_close** (object link_identifier)

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplo 1. dbx_close() example

```
<?php
$link = dbx_connect(DBX_MYSQL, "localhost", "db", "username", "password")
    or die("Could not connect");

echo "Connected successfully";
dbx_close($link);
?>
```

Nota: Always refer to the module-specific documentation as well.

See also **dbx_connect()**.

dbx_compare

(PHP 4 >= 4.1.0, PHP 5)

dbx_compare -- Compare two rows for sorting purposes

Description

int **dbx_compare** (array row_a, array row_b, string column_key [, int flags])

dbx_compare() returns *0* if the *row_a[$column_key]* is equal to *row_b[$column_key]*, and *1* or *-1* if the former is greater or is smaller than the latter one, respectively, or vice versa if the *flag* is set to **DBX_CMP_DESC**. **dbx_compare()** is a helper function for **dbx_sort()** to ease the make and use of the custom sorting function.

The *flags* can be set to specify comparison direction:

- **DBX_CMP_ASC** - ascending order
- **DBX_CMP_DESC** - descending order

and the preferred comparison type:

- **DBX_CMP_NATIVE** - no type conversion
- **DBX_CMP_TEXT** - compare items as strings
- **DBX_CMP_NUMBER** - compare items numerically

One of the direction and one of the type constant can be combined with bitwise OR operator (|). The default value for the *flags* parameter is **DBX_CMP_ASC** | **DBX_CMP_NATIVE**.

Ejemplo 1. dbx_compare() example

```
<?php
function user_re_order($a, $b)
{
    $rv = dbx_compare($a, $b, "parentid", DBX_CMP_DESC);
    if (!$rv) {
        $rv = dbx_compare($a, $b, "id", DBX_CMP_NUMBER);
    }
    return $rv;
}

$link   = dbx_connect(DBX_ODBC, "", "db", "username", "password")
    or die("Could not connect");

$result = dbx_query($link, "SELECT id, parentid, description FROM table ORDER BY id");
    // data in $result is now ordered by id

dbx_sort($result, "user_re_order");
    // date in $result is now ordered by parentid (descending), then by id

dbx_close($link);
?>
```

See also **dbx_sort()**.

dbx_connect

(PHP 4 >= 4.0.6, PHP 5)

dbx_connect -- Open a connection/database

Description

object **dbx_connect** (mixed module, string host, string database, string username, string password [, int persistent])

dbx_connect() returns an object on success, **FALSE** on error. If a connection has been made but the database could not be selected, the connection is closed and **FALSE** is returned. The *persistent* parameter can be set to **DBX_PERSISTENT**, if so, a persistent connection will be created.

The *module* parameter can be either a string or a constant, though the latter form is preferred. The possible values are given below, but keep in mind that they only work if the module is actually loaded.

- **DBX_MYSQL** or "mysql"
- **DBX_ODBC** or "odbc"
- **DBX_PGSQL** or "pgsql"
- **DBX_MSSQL** or "mssql"
- **DBX_FBSQL** or "fbsql" (available from PHP 4.1.0)
- **DBX_SYBASECT** or "sybase_ct" (available from PHP 4.2.0)
- **DBX_OCI8** or "oci8" (available from PHP 4.3.0)

- **DBX_SQLITE** or "sqlite" (PHP 5)

The *host*, *database*, *username* and *password* parameters are expected, but not always used depending on the connect functions for the abstracted module.

The returned *object* has three properties:

database

It is the name of the currently selected database.

handle

It is a valid handle for the connected database, and as such it can be used in module-specific functions (if required).

```
<?php
$link = dbx_connect(DBX_MYSQL, "localhost", "db", "username", "password");
mysql_close($link->handle); // dbx_close($link) would be better here
?>
```

module

It is used internally by dbx only, and is actually the module number mentioned above.

Ejemplo 1. dbx_connect() example

```
<?php
$link = dbx_connect(DBX_ODBC, "", "db", "username", "password", DBX_PERSISTENT)
    or die("Could not connect");

echo "Connected successfully";
dbx_close($link);
?>
```

Nota: Always refer to the module-specific documentation as well.

See also **dbx_close()**.

dbx_error

(PHP 4 >= 4.0.6, PHP 5)

dbx_error -- Report the error message of the latest function call in the module (not just in the connection)

Description

string **dbx_error** (object link_identifier)

dbx_error() returns a string containing the error message from the last function call of the abstracted module (e.g. mysql module). If there are multiple connections in the same module, just the last error is given. If there are connections on different modules, the latest error is returned for the module specified by the *link_identifier* parameter.

Ejemplo 1. dbx_error() example

```
<?php
$link   = dbx_connect(DBX_MYSQL, "localhost", "db", "username", "password")
    or die("Could not connect");

$result = dbx_query($link, "select id from non_existing_table");
if ($result == 0) {
    echo dbx_error($link);
}
dbx_close($link);
?>
```

Nota: Always refer to the module-specific documentation as well.

The error message for Microsoft SQL Server is actually the result of the **mssql_get_last_message()** function.

The error message for Oracle (oci8) is not implemented (yet).

dbx_escape_string

(PHP 4 >= 4.3.0, PHP 5)

dbx_escape_string -- Escape a string so it can safely be used in an sql-statement

Description

string **dbx_escape_string** (object link_identifier, string text)

dbx_escape_string() returns the text, escaped where necessary (such as quotes, backslashes etc). It returns NULL on error.

Ejemplo 1. dbx_escape_string() example

```
<?php
$link   = dbx_connect(DBX_MYSQL, "localhost", "db", "username", "password")
    or die("Could not connect");

$text = dbx_escape_string($link, "It\'s quoted and backslashed (\\).");
$result = dbx_query($link, "insert into tbl (txt) values ('" . $text . "')");
if ($result == 0) {
    echo dbx_error($link);
}
dbx_close($link);
?>
```

See also **dbx_query()**.

dbx_fetch_row

(PHP 5)

dbx_fetch_row -- Fetches rows from a query-result that had the **DBX_RESULT_UNBUFFERED** flag set

Description

object **dbx_fetch_row** (object result_identifier)

dbx_fetch_row() returns a row on success, and *0* on failure (e.g. when no more rows are available). When the **DBX_RESULT_UNBUFFERED** is not set in the query, **dbx_fetch_row()** will fail as all rows have already been fetched into the results data property.

As a side effect, the rows property of the query-result object is incremented for each successful call to **dbx_fetch_row()**.

Ejemplo 1. How to handle the returned value

```
<?php
$result = dbx_query($link, 'SELECT id, parentid, description FROM table', DBX_RESULT_UN

echo "<table>\n";
while ($row = dbx_fetch_row($result)) {
    echo "<tr>\n";
    foreach ($row as $field) {
        echo "<td>$field</td>";
    }
    echo "</tr>\n";
}
echo "</table>\n";
?>
```

The *result_identifier* parameter is the result object returned by a call to **dbx_query()**.

The returned array contains the same information as any row would have in the dbx_query result data property, including columns accessible by index or fieldname when the flags for dbx_guery were set that way.

See also **dbx_query()**.

dbx_query

(PHP 4 >= 4.0.6, PHP 5)

dbx_query -- Send a query and fetch all results (if any)

Description

object **dbx_query** (object link_identifier, string sql_statement [, int flags])

dbx_query() returns an object or *1* on success, and *0* on failure. The result object is returned only if the query given in *sql_statement* produces a result set (i.e. a SELECT query, even if the result set is empty).

Ejemplo 1. How to handle the returned value

```
<?php
$link   = dbx_connect(DBX_ODBC, "", "db", "username", "password")
    or die("Could not connect");

$result = dbx_query($link, 'SELECT id, parentid, description FROM table');

if (is_object($result) ) {
    // ... do some stuff here, see detailed examples below ...
    // first, print out field names and types
    // then, draw a table filled with the returned field values
} else {
    exit("Query failed");
}

dbx_close($link);
?>
```

The *flags* parameter is used to control the amount of information that is returned. It may be any combination of the following constants with the bitwise OR operator (|). The DBX_COLNAMES_* flags override the dbx.colnames_case setting from `php.ini`.

DBX_RESULT_INDEX

It is *always* set, that is, the returned object has a data property which is a 2 dimensional array indexed numerically. For example, in the expression *data[2][3] 2* stands for the row (or record) number and *3* stands for the column (or field) number. The first row and column are indexed at 0.

If **DBX_RESULT_ASSOC** is also specified, the returning object contains the information related to **DBX_RESULT_INFO** too, even if it was not specified.

DBX_RESULT_INFO

It provides info about columns, such as field names and field types.

DBX_RESULT_ASSOC

It effects that the field values can be accessed with the respective column names used as keys to the returned object's data property.

Associated results are actually references to the numerically indexed data, so modifying *data [0][0]* causes that *data[0]['field_name_for_first_column']* is modified as well.

DBX_RESULT_UNBUFFERED (PHP 5)

This flag will not create the data property, and the rows property will initially be 0. Use this flag for large datasets, and use **dbx_fetch_row()** to retrieve the results row by row.

The **dbx_fetch_row()** function will return rows that are conformant to the flags set with this query. Incidentally, it will also update the rows each time it is called.

DBX_COLNAMES_UNCHANGED (available from PHP 4.3.0)

The case of the returned column names will not be changed.

DBX_COLNAMES_UPPERCASE (available from PHP 4.3.0)

The case of the returned column names will be changed to uppercase.

DBX_COLNAMES_LOWERCASE (available from PHP 4.3.0)

The case of the returned column names will be changed to lowercase.

Note that **DBX_RESULT_INDEX** is always used, regardless of the actual value of *flags* parameter. This means that only the following combinations are effective:

- **DBX_RESULT_INDEX**
- **DBX_RESULT_INDEX** | **DBX_RESULT_INFO**
- **DBX_RESULT_INDEX** | **DBX_RESULT_INFO** | **DBX_RESULT_ASSOC** - this is the default, if *flags* is not specified.

The returned *object* has four or five properties depending on *flags*:

handle

It is a valid handle for the connected database, and as such it can be used in module specific functions (if required).

```
<?php
$result = dbx_query($link, "SELECT id FROM table");
mysql_field_len($result->handle, 0);
?>
```

cols and rows

These contain the number of columns (or fields) and rows (or records) respectively.

```
<?php
$result = dbx_query($link, 'SELECT id FROM table');
echo $result->rows; // number of records
echo $result->cols; // number of fields
?>
```

info (optional)

It is returned only if either **DBX_RESULT_INFO** or **DBX_RESULT_ASSOC** is specified in the *flags* parameter. It is a 2 dimensional array, that has two named rows (*name* and *type*) to retrieve column information.

Ejemplo 2. lists each field's name and type

```
<?php
$result = dbx_query($link, 'SELECT id FROM table',
                      DBX_RESULT_INDEX | DBX_RESULT_INFO);

for ($i = 0; $i < $result->cols; $i++ ) {
    echo $result->info['name'][$i] . "\n";
    echo $result->info['type'][$i] . "\n";
}
?>
```

data

This property contains the actual resulting data, possibly associated with column names as well depending on *flags*. If **DBX_RESULT_ASSOC** is set, it is possible to use *$result->data [2]["field_name"]*.

Ejemplo 3. outputs the content of data property into HTML table

```
<?php
$result = dbx_query($link, 'SELECT id, parentid, description FROM table');

echo "<table>\n";
foreach ($result->data as $row) {
    echo "<tr>\n";
    foreach ($row as $field) {
        echo "<td>$field</td>";
    }
    echo "</tr>\n";
}
echo "</table>\n";
?>
```

Ejemplo 4. How to handle UNBUFFERED queries

```
<?php

$result = dbx_query ($link, 'SELECT id, parentid, description FROM table', DBX_RES

echo "<table>\n";
while ($row = dbx_fetch_row($result)) {
    echo "<tr>\n";
    foreach ($row as $field) {
        echo "<td>$field</td>";
    }
    echo "</tr>\n";
}
echo "</table>\n";

?>
```

Nota: Always refer to the module-specific documentation as well.

Column names for queries on an Oracle database are returned in lowercase.

See also **dbx_escape_string()**, **dbx_fetch_row()** and **dbx_connect()**.

dbx_sort

(PHP 4 >= 4.0.6, PHP 5)

dbx_sort -- Sort a result from a dbx_query by a custom sort function

Description

bool **dbx_sort** (object result, string user_compare_function)

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Nota: It is always better to use *ORDER BY SQL* clause instead of **dbx_sort()**, if possible.

Ejemplo 1. dbx_sort() example

```
<?php
function user_re_order($a, $b)
{
    $rv = dbx_compare($a, $b, "parentid", DBX_CMP_DESC);
    if (!$rv) {
        $rv = dbx_compare($a, $b, "id", DBX_CMP_NUMBER);
    }
    return $rv;
}

$link   = dbx_connect(DBX_ODBC, "", "db", "username", "password")
    or die("Could not connect");

$result = dbx_query($link, "SELECT id, parentid, description FROM tbl ORDER BY id");
    // data in $result is now ordered by id

dbx_sort($result, "user_re_order");
    // data in $result is now ordered by parentid (descending), then by id

dbx_close($link);
?>
```

See also **dbx_compare()**.

XXV. Funciones de acceso directo a E/S

Introducción

PHP incluye soporte para funciones de acceso directo a E/S tal y como se especifican en la sección sexta del estándar de Posix. Estas funciones permiten realizar operaciones de E/S a un nivel inferior al de las funciones genéricas de C como **fopen()** y **fread()**. Las funciones de acceso directo a E/S solo deberían emplearse cuando se requiere un control directo de un determinado dispositivo. En todos los demás casos, es más adecuado el empleo de las funciones estándar del sistema de archivos.

> **Nota:** Esta extensión no está disponible en plataformas Windows

---

Requirimientos

No se necesitan bibliotecas externas para construir esta extensión

---

Instalación

Para usar las funciones de acceso directo a E/S, se debe añadir el parámetro *--enable-dio* a las opciones de configuración de PHP.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

La extensión de las funciones de acceso directo a E/S define un nuevo tipo de recurso: un descriptor de archivo devuelto por la función **dio_open()**.

Constantes predefinidas

Esta extensión no tiene ninguna constante definida.

Tabla de contenidos

dio_close

(PHP 4 >= 4.2.0, PHP 5)

dio_close -- Cierra el descriptor de archivo indicado por el parámetro fd

Descripción

void **dio_close** (resource fd)

La función **dio_close()** cierra el descriptor de archivo indicado por el parámetro *fd*.

dio_fcntl

(PHP 4 >= 4.2.0, PHP 5)

dio_fcntl -- Realiza una operación del tipo fcntl de la libería de C sobre el descriptor de archivo indicado por el parámetro fd

Descripción

mixed **dio_fcntl** (resource fd, int cmd [, mixed arg])

La función **dio_fcntl()** realiza la operación indicada por el parámetro *cmd* sobre el descriptor de archivo *fd*. Algunas operaciones requieren argumentos adicionales introducidos a traves del

parámetro *args*.

arg es un array de tipo asociativo siempre que *cmd* es F_SETLK o F_SETLLW. En ese caso, las claves del array son las siguientes:

- "start" - lugar donde comienza el bloqueo (lock)
- "length" - tamaño del area bloqueada. Un valor igual a 0 (cero) indica un bloqueo hasta el final del archivo
- "wenth" - Lugar desde donde se empieza a contar el tamaño l_start. Puede ser SEEK_SET, SEEK_END y SEEK_CUR
- "type" - tipo de bloqueo. Puede ser F_RDLCK (bloqueo de lectura), F_WRLCK (bloqueo de escritura) o F_UNLCK (desbloqueo)

cmd puede ser cualquiera de las siguientes operaciones:

- F_SETLK - Activa o desactiva el bloqueo. Si alguien lo ha bloqueado anteriormente, la función **dio_fcntl()** devuelve -1.
- F_SETLKW - Es similar a F_SETLK, salvo que en esta ocasión la función **dio_fcntl()** espera a que se libere el bloqueo en caso de que alguien lo haya bloqueado anteriormente.
- F_GETLK - La función **dio_fcntl()** devuelve un array asociativo como el descrito previamente si existe alguien que impide realizar un bloqueo. Si no existe un bloqueo anterior, la clave "type" se pondra a F_UNLCK.
- F_DUPFD - Busca el descriptor de archivo disponible con el número mas bajo y que sea mayor o igual que *arg* y lo convierte en una copia de *fd* y devuelve los dos descriptores.
- F_SETFL - Establece el valor de las opciones de los descriptores de archivos a los valores especificados en el parámetro *arg*. Las opciones pueden ser O_APPEND,O_NONBLOCK o O_ASYNC. Para poder utilizar la opción O_ASYNC es necesario utilizar la extensión PCNTL.

dio_open

(PHP 4 >= 4.2.0, PHP 5)

dio_open -- Abre un archivo cuyo nombre indica el parámetro "nombre_archivo" con las opciones indicadas por "flags" y los permisos establecidos con "modo"

Descripción

resource **dio_open** (string nombre_archivo, int flags [, int modo])

La función **dio_open()** abre un archivo y devuelve un descriptor para su manejo o **`FALSE`** si ha ocurrido algun error. Si el valor de *flags* es O_CREAT, el tercer parámetro *modo* establecerá los permisos de creación del archivo. El parámetro *flags* puede tener uno de los siguientes valores:

- O_RDONLY - abre el archivo para acceso de solo lectura.

- O_WRONLY - abre el archivo para acceso de solo escritura.
- O_RDWR - abre el archivo para acceso tanto de lectura como de escritura.

El parámetro *flags* puede incluir tambien cualquier combinación de los siguientes valores:

- O_CREAT - crea el archivo si no existía previamente.
- O_EXCL - si se indican de forma simultanea los valores O_CREAT y O_EXCL, la función **dio_open()** falla si el archivo existía previamente.
- O_TRUNC - si el archivo existe y se permiten las operaciones de escritura sobre el, se elimina todo el contenido anterior del archivo y su tamaño se pone a cero.
- O_APPEND - las operaciones de escritura sobre el archivo escriben los datos al final del archivo.
- O_NONBLOCK - el archivo se pone en modo no-bloqueante.

dio_read

(PHP 4 >= 4.2.0, PHP 5)

dio_read -- Lee hasta *n* bytes del archivo cuyo descriptor es *fd* y los devuelve. Si no se le indica el valor de *n*, se leen hasta 1024 bytes.

Descripción

string **dio_read** (resource fd [, int n])

La función **dio_read()** lee y devuelve *n* bytes del archivo cuyo descriptor es *fd*. Si no se especifica el valor del parámetro *n*, **dio_read()** lee hasta 1024 bytes y los devuelve.

dio_seek

(PHP 4 >= 4.2.0, PHP 5)

dio_seek -- Cambia el posicionamiento en el archivo cuyo descriptor es *fd* a traves de los parámetros *pos* y *whence*

Descripción

int **dio_seek** (resource fd, int pos, int whence)

La función **dio_seek()** se utiliza para cambiar el posicionamiento en el archivo cuyo descriptor es *fd*. El parámetro *whence* indica la forma en que debe ser interpretado el parámetro *pos*:

- SEEK_SET - indica que se deben avanzar *pos* bytes desde el comienzo del archivo.
- SEEK_CUR - indica que se deben avanzar *pos* bytes a partir de la posición actual. El avance puede ser tanto positivo como negativo.

- SEEK_END - indica que se deben avanzar *pos* bytes desde el final del archivo. Un avance negativo implica una posición dentro de la longitud original del archivo y un avance positivo implica una posición más alla de la longitud original del archivo. En este último caso en el que la posición se encuentra fuera de los limites del archivo original, si se realiza una operación de escritura, las posiciones comprendidas entre el final del archivo original y la posición de comienzo de los nuevos datos, se rellenarán con ceros.

dio_stat

(PHP 4 >= 4.2.0, PHP 5)

dio_stat -- Obtiene la información sobre el archivo cuyo descriptor es *fd*

Descripción

array **dio_stat** (resource fd)

La función **dio_stat()** devuelve la información sobre el archivo cuyo descriptor es *fd*. El valor devuelto por **dio_stat()** es un array asociativo con las siguientes claves:

- "device" - dispositivo
- "inode" - inodo
- "mode" - modo
- "nlink" - numero de "hard links"
- "uid" - identificador de usuario
- "gid" - identificador de grupo
- "device_type" - tipo de dispositivo (si es un dispositivo con inodos)
- "size" - tamaño total en bytes
- "blocksize" - tamaño de bloque
- "blocks" - numero de bloques ocupados
- "atime" - hora del ultimo acceso
- "mtime" - hora de la ultima modificacion
- "ctime" - hora del ultimo cambio

Si se produce un error, la función **dio_stat()** devuelve un valor NULL.

dio_tcsetattr

(PHP 4 >= 4.3.0, PHP 5)

dio_tcsetattr -- Configura las opciones de un terminal y la velocidad de un puerto serie

Descripción

dio_tcsetattr (resource fd, array options)

La función **dio_tcsetattr()** configura las opciones de un terminal y la velocidad en baudios del recurso *resource* abierto previamente. Las opciones actualmente disponibles son las siguientes:

- 'baud' - velocidad en baudios del puerto serie - puede tomar los valores 38400, 19200, 9600, 4800, 2400, 1800, 1200, 600, 300, 200, 150, 134, 110, 75 o 50. Por defecto se establece una velocidad de 9600 baudios.
- 'bits' - número de bits de los datos - puede ser 8, 7, 6 o 5. Por defecto se establece un valor de 8 bits.
- 'stop' - bits de stop - pueden ser 1 o 2. Por defecto se establece un valor de 1 bit de stop.
- 'parity' - bits de paridad - pueden ser 0, 1 o 2. Por defecto se establece un valor de 0 bits de paridad.

Ejemplo 1. Configurar la velocidad de un puerto serie

```
<?php

$fd = dio_open('/dev/ttyS0', O_RDWR | O_NOCTTY | O_NONBLOCK);

dio_fcntl($fd,F_SETFL, O_SYNC );

dio_tcsetattr($fd, array(
  'baud' => 9600,
  'bits' => 8,
  'stop'  =>1,
  'parity' => 0
));

while (1) {

  $data = dio_read($fd,256);

  if ($data) {
      echo $data;
  }
}

?>
```

dio_truncate

(PHP 4 >= 4.2.0, PHP 5)

dio_truncate -- Trunca el tamaño del archivo cuyo descriptor es *fd* hasta un valor de *offset* bytes

Descripción

bool **dio_truncate** (resource fd, int offset)

La función **dio_truncate()** trunca el archivo cuyo descriptor es *fd* y cambia su tamaño hasta un

valor de *offset* bytes. Si la longitud del archivo original era mayor que el valor indicado, el archivo se trunca y se pierde toda la información restante. Si el valor indicado es mayor que la longitud original del archivo, no se especifica si el archivo permanece igual o se extiende su longitud hasta llegar al valor indicado por *offset* rellenando con ceros los nuevos bytes del archivo. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo..

dio_write

(PHP 4 >= 4.2.0, PHP 5)

dio_write -- Escribe datos en el archivo cuyo descriptor es *fd*

Descripción

int **dio_write** (resource fd, string data [, int len])

La función **dio_write()** escribe hasta *len* bytes de los datos indicados por el parámetro *data* en el archivo cuyo descriptor es *fd*. Si no se especifica el valor de *len*, la función **dio_write()** escribe todos los datos del parámetro *data* en el archivo indicado. La función **dio_write()** devuelve el numero de bytes escritos en el archivo cuyo descriptor es *fd*.

XXVI. Funciones de Directorio

Introducción

Requirimientos

No se necesitan bibliotecas externas para construir esta extensión

---

Instalación

No se necesita ninguna instalación para usar estas funciones, son parte del núcleo de PHP.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

`DIRECTORY_SEPARATOR` (**string**)

`PATH_SEPARATOR` (**string**)

> **Nota:** El parámetro **`PATH_SEPARATOR`** fue introducido con PHP 4.3.0-RC2.

Ver también

Para consultar sobre funciones relacionadas como **dirname()**, **is_dir()**, **mkdir()**, y **rmdir()**, vea la sección Sistema de archivos.

Tabla de contenidos

chdir

(PHP 3, PHP 4 , PHP 5)

chdir -- Cambiar de directorio

Descripción

bool **chdir** (string directorio)

Cambiar el directorio actual de PHP a *directorio*. Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Ejemplo 1. Ejemplo de chdir()

```
<?php

// directorio actual
echo getcwd() . "\n";

chdir('public_html');

// directorio actual
echo getcwd() . "\n";

?>
```

Este ejemplo producirá la salida:

```
/home/vincent
/home/vincent/public_html
```

> **Nota:** Cuando safe-mode (modo-seguro) está activado, PHP comprueba si los directorios que va a utilizar tienen la misma UID que el script que está siendo ejecutado.

Vea también **getcwd()**.

chroot

(PHP 4 >= 4.0.5, PHP 5)

chroot -- Cambiar el directorio raíz

Descripción

bool **chroot** (string directorio)

Cambia el directorio raíz del proceso actual a *directorio*. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Esta función sólo se encuentra disponible si su sistema la soporta y se encuentra usando un entorno CLI, CGI o SAPI embebido.

> **Nota: chroot()** requiere privilegios de root.

> **Nota:** Esta función no está implementada en plataformas Windows.

dir

(PHP 3, PHP 4 , PHP 5)

dir -- clase directorio

Descripcion

new **dir** (string directory)

Un mecanismo semi-orientado a objetos para leer directorios. El parametro *directory* abre el directorio. Dos propiedades estan disponibles cuando el directorio ha sido abierto. La propiedad de manejo puede ser usada con otras funciones de directorios tal como **readdir()**, **rewinddir()** y **closedir()**. La propiedad de trayectoria (path) es fijada para encaminar el directorio que ha sido

abierto. Tres metodos estan disponibles: leer, rebobinar y cerrar.

Ejemplo 1. dir() Ejemplo

```
$d = dir("/etc");
      echo "Handle: ".$d->handle."<br>\n";
      echo "Path: ".$d->path."<br>\n";
      while($entry=$d->read()) {
      echo $entry."<br>\n";
      }
      $d->close();
```

closedir

(PHP 3, PHP 4 , PHP 5)

closedir -- cierra el manejador de directorios

Descripcion

void **closedir** (int dir_handle)

Cierra la secuencia de directorio determinada por *dir_handle*. La secuencia debe de haber sido abierta previamente con **opendir()**.

getcwd

(PHP 4 , PHP 5)

getcwd -- Obtiene el directorio de trabajo actual

Descripción

string **getcwd** (void)

Devuelve el directorio de trabajo actual, o **FALSE** en caso de fallo.

> **Nota:** En algunas variantes de Unix, **getcwd()** devolverá **FALSE** si alguno de los directorios padre no tiene definido el modo de lectura o búsqueda, incluso si el directorio actual lo tiene. Vea **chmod()** para más información sobre los modos y permisos.

Ejemplo 1. Ejemplo de getcwd()

```
<?php

// directorio actual
echo getcwd() . "\n";

chdir('cvs');

// directorio actual
echo getcwd() . "\n";

?>
```

El resultado del ejemplo seria algo similar a:

```
/home/didou
/home/didou/cvs
```

Vea también **chdir()** y **chmod()**.

opendir

(PHP 3, PHP 4 , PHP 5)

opendir -- abre el manejador de directorios

Descripcion

int **opendir** (string path)

Devuelve un manejador de directorio para ser usado con las llamadas **closedir()**, **readdir()** y **rewinddir()**.

readdir

(PHP 3, PHP 4 , PHP 5)

readdir -- lee las entradas del manejador de directorios

Descripcion

string **readdir** (int dir_handle)

Devuelve el nombre del siguiente fichero en el directorio. Los nombres de ficheros no son devueltos en ningun orden especial .

Ejemplo 1. Listar todos los ficheros en un directorio

```
<?php
      $handle=opendir('.');
      echo "Directory handle: $handle\n";
      echo "Files:\n";
      while ($file = readdir($handle)) {
       echo "$file\n";
       }
       closedir($handle);
       ?>
```

Tener en cuenta que **readdir()** devolvera tambien . y .. Si no quereis estas entradas podeis borrarlas:

Ejemplo 2. Listar todos los ficheros en un directorio excepto . y ..

```
<?php
if ($handle = opendir('.')) {
    while (false !== ($file = readdir($handle))) {
        if ($file != "." && $file != "..") {
            echo "$file\n";
        }
    }
    closedir($handle);
}
?>
```

rewinddir

(PHP 3, PHP 4 , PHP 5)

rewinddir -- rebobinar el manejador de directorios

Descripcion

void **rewinddir** (int dir_handle)

Inizializa la secuencia de directorio determinada por *dir_handle* al principio del directorio.

scandir

(PHP 5)

scandir -- Lista los archivos y directorios ubicados en la ruta especificada

Descripción

array **scandir** (string directorio [, int sentido_de_ordenamiento [, resource contexto]])

Devuelve un **array** de archivos y directorios que se encuentran bajo *directorio*. Si *directorio* no es un directorio, entonces el valor booleano **FALSE** es retornado, y se genera un error de nivel **E_WARNING**.

Por defecto, el sentido del ordenamiento es ascendente. Si es usado el parámetro opcional *sentido_de_ordenamiento* (definido como 1), entonces el sentido será descendente.

Ejemplo 1. Un ejemplo simple de scandir()

```
<?php
$dir    = '/tmp';
$archivos1 = scandir($dir);
$archivos2 = scandir($dir, 1);

print_r($archivos1);
print_r($archivos2);
?>
```

Genera una salida como:

```
Array
(
    [0] => .
    [1] => ..
    [2] => bar.php
    [3] => directorio_cualquiera
    [4] => foo.txt
)
Array
(
    [0] => foo.txt
    [1] => directorio_cualquiera
    [2] => bar.php
    [3] => ..
    [4] => .
)
```

Ejemplo 2. Alternativas a scandir() con PHP 4

```
<?php
$dir = "/tmp";
$dh  = opendir($dir);
while (false !== ($nombre_archivo = readdir($dh))) {
    $archivos[] = $nombre_archivo;
}

sort($archivos);

print_r($archivos);

rsort($archivos);

print_r($archivos);

?>
```

Genera una salida como:

```
Array
(
    [0] => .
    [1] => ..
    [2] => bar.php
    [3] => directorio_cualquiera
    [4] => foo.txt
)
Array
(
    [0] => foo.txt
    [1] => directorio_cualquiera
    [2] => bar.php
    [3] => ..
    [4] => .
)
```

> **Sugerencia:** Puede usar una URL como nombre de archivo con esta función si los fopen wrappers han sido activados. Consulte **fopen()** para más detalles sobre cómo especificar el nombre de fichero y Apéndice L una lista de protocolos URL soportados

Vea también **opendir()**, **readdir()**, **glob()**, **is_dir()**, y **sort()**.

XXVII. DOM Functions

Introducción

The DOM extension is the replacement for the DOM XML extension from PHP 4. The extension still contains many old functions, but they should no longer be used. In particular, functions that are not object-oriented should be avoided.

The extension allows you to operate on an XML document with the DOM API.

Instalación

No se necesita ninguna instalación para usar estas funciones, son parte del núcleo de PHP.

Clases predefinidas

The API of the module follows the DOM Level 2 standard as closely as possible. Consequently, the API is fully object-oriented. It is a good idea to have the DOM standard available when using this module.

This module defines a number of classes, which are explained in the following tables. Classes with an ñalent in the DOM standard are named DOMxxx.

DOMAttr

Extends **DOMNode**.

Métodos

- DOMAttr->isId() - Checks if attribute is a defined ID

Propiedades

Tabla 1.

Name	Type	Read-only	Description
name	string	yes	The name of the attribute
ownerElement	DOMElement	yes	The element which contains the attribute
schemaTypeInfo	bool	yes	Not implemented yet, always return **TRUE**
specified	bool	yes	Not implemented yet, always return **TRUE**
value	string	no	The value of the attribute

DOMCharacterData

Extends **DOMNode**.

Métodos

- DOMCharacterData->appendData() - Append a string to the end of the character data of the node
- DOMCharacterData->deleteData() - Remove a range of characters from the node
- DOMCharacterData->insertData() - Insert a string at the specified 16-bit unit offset

- DOMCharacterData->replaceData() - Replace a substring within the DOMCharacterData node
- DOMCharacterData->substringData() - Extracts a range of data from the node

---

Propiedades

Tabla 2.

Name	Type	Read-only	Description
data	string	no	The contents of the node
length	int	yes	The length of the contents

DOMDocument

Extends **DOMNode**.

Constructor

- DOMDocument->__construct() - construct a new DOMDocument object

---

Métodos

- DOMDocument->createAttribute() - Create new attribute
- DOMDocument->createAttributeNS() - Create new attribute node with an associated namespace
- DOMDocument->createCDATASection() - Create new cdata node
- DOMDocument->createComment() - Create new comment node
- DOMDocument->createDocumentFragment() - Create new document fragment
- DOMDocument->createElement() - Create new element node
- DOMDocument->createElementNS() - Create new element node with an associated namespace
- DOMDocument->createEntityReference() - Create new entity reference node
- DOMDocument->createProcessingInstruction() - Creates new PI node
- DOMDocument->createTextNode() - Create new text node

- DOMDocument->getElementById() - Searches for an element with a certain id
- DOMDocument->getElementsByTagName() - Searches for all elements with given tag name
- DOMDocument->getElementsByTagNameNS() - Searches for all elements with given tag name in specified namespace
- DOMDocument->importNode() - Import node into current document
- DOMDocument->load() - Load XML from a file
- DOMDocument->loadHTML() - Load HTML from a string
- DOMDocument->loadHTMLFile() - Load HTML from a file
- DOMDocument->loadXML() - Load XML from a string
- DOMDocument->normalize() - Normalizes document
- DOMDocument->relaxNGValidate() - Performs relaxNG validation on the document
- DOMDocument->relaxNGValidateSource() - Performs relaxNG validation on the document
- DOMDocument->save() - Dumps the internal XML tree back into a file
- DOMDocument->saveHTML() - Dumps the internal document into a string using HTML formatting
- DOMDocument->saveHTMLFile() - Dumps the internal document back into a file using HTML formatting
- DOMDocument->saveXML() - Dumps the internal XML tree back into a string
- DOMDocument->schemaValidate() - Validates a document based on a schema
- DOMDocument->schemaValidateSource() - Validates a document based on a schema
- DOMDocument->validate() - Validates the document based on its DTD
- DOMDocument->xinclude() - Substitutes XIncludes in a DOMDocument Object

---

Propiedades

Tabla 3.

Name	Type	Read-only	Description
actualEncoding	string	yes	
config	DOMConfiguration	yes	
doctype	DOMDocumentType	yes	The Document Type Declaration associated with this document.

Name	Type	Read-only	Description
documentElement	DOMElement	yes	This is a convenience attribute that allows direct access to the child node that is the document element of the document.
documentURI	string	no	The location of the document or **NULL** if undefined.
encoding	string	no	
formatOutput	bool	no	
implementation	DOMImplementation	yes	The **DOMImplementation** object that handles this document.
preserveWhiteSpace	bool	no	Do not remove redundant white space. Default to **TRUE**.
recover	bool	no	
resolveExternals	bool	no	Set it to **TRUE** to load external entities from a doctype declaration. This is useful for including character entities in your XML document.
standalone	bool	no	
strictErrorChecking	bool	no	Throws DOMException on errors. Default to **TRUE**.
substituteEntities	bool	no	
validateOnParse	bool	no	Loads and validates against the DTD. Default to **FALSE**.
version	string	no	
xmlEncoding	string	yes	An attribute specifying, as part of the XML declaration, the encoding of this document. This is **NULL** when unspecified or when it is not known, such as when the Document was created in memory.
xmlStandalone	bool	no	An attribute specifying, as part of the XML declaration, whether this document is standalone. This is **FALSE** when unspecified.
xmlVersion	string	no	An attribute specifying, as part of the XML declaration, the version number of this document. If there is no declaration and if this document supports the "XML" feature, the value is "1.0".

DOMDocumentType

Extends **DOMNode**

Each **DOMDocument** has a *doctype* attribute whose value is either **NULL** or a **DOMDocumentType** object.

Propiedades

Tabla 4.

Name	Type	Read-only	Description
publicId	string	yes	The public identifier of the external subset.
systemId	string	yes	The system identifier of the external subset. This may be an absolute URI or not.
name	string	yes	The name of DTD; i.e., the name immediately following the *DOCTYPE* keyword.
entities	DOMNamedNodeMap	yes	A **DOMNamedNodeMap** containing the general entities, both external and internal, declared in the DTD.
notations	DOMNamedNodeMap	yes	A **DOMNamedNodeMap** containing the notations declared in the DTD.
internalSubset	string	yes	The internal subset as a string, or null if there is none. This is does not contain the delimiting square brackets.

DOMElement

Extends **DOMNode**.

Métodos

- DOMElement->getAttribute() - Returns value of attribute
- DOMElement->getAttributeNode() - Returns attribute node
- DOMElement->getAttributeNodeNS() - Returns attribute node
- DOMElement->getAttributeNS() - Returns value of attribute
- DOMElement->getElementsByTagName() - Gets elements by tagname
- DOMElement->getElementsByTagNameNS() - Get elements by namespaceURI and localName
- DOMElement->hasAttribute() - Checks to see if attribute exists
- DOMElement->hasAttributeNS() - Checks to see if attribute exists
- DOMElement->removeAttribute() - Removes attribute
- DOMElement->removeAttributeNode() - Removes attribute
- DOMElement->removeAttributeNS() - Removes attribute

- DOMElement->setAttribute() - Adds new attribute
- DOMElement->setAttributeNode() - Adds new attribute node to element
- DOMElement->setAttributeNodeNS() - Adds new attribute node to element
- DOMElement->setAttributeNS() - Adds new attribute

Propiedades

Tabla 5.

Name	Type	Read-only	Description
schemaTypeInfo	bool	yes	Not implemented yet, always return **TRUE**
tagName	string	yes	The element name

DOMEntity

Extends **DOMNode**

This interface represents a known entity, either parsed or unparsed, in an XML document.

Propiedades

Tabla 6.

Name	Type	Read-only	Description
publicId	string	yes	The public identifier associated with the entity if specified, and **NULL** otherwise.
systemId	string	yes	The system identifier associated with the entity if specified, and **NULL** otherwise. This may be an absolute URI or not.
notationName	string	yes	For unparsed entities, the name of the notation for the entity. For parsed entities, this is **NULL**.
actualEncoding	string	no	An attribute specifying the encoding used for this entity at the time of parsing, when it is an external parsed entity. This is **NULL** if it an entity from the internal subset or if it is not known.
encoding	string	yes	An attribute specifying, as part of the text declaration, the encoding of this entity, when it is an external parsed entity. This is **NULL** otherwise.
version	string	yes	An attribute specifying, as part of the text declaration, the version number of this entity, when it is an external parsed entity. This is **NULL** otherwise.

DOMEntityReference

Extends **DOMNode**.

DOMException

DOM operations raise exceptions under particular circumstances, i.e., when an operation is impossible to perform for logical reasons.

See also Capítulo 20.

Propiedades

Tabla 7.

Name	Type	Read-only	Description
code	int	yes	An integer indicating the type of error generated

---

DOMImplementation

The **DOMImplementation** interface provides a number of methods for performing operations that are independent of any particular instance of the document object model.

Métodos

- DOMImplementation->createDocument() - Creates a DOM Document object of the specified type with its document element
- DOMImplementation->createDocumentType() - Creates an empty DOMDocumentType object
- DOMImplementation->hasFeature() - Test if the DOM implementation implements a specific feature

---

DOMNameList

Propiedades

Tabla 8.

Name	Type	Read-only	Description
length	int	yes	The number of pairs (name and namespaceURI) in the list. The range of valid child node indices is 0 to *length - 1* inclusive.

DOMNode

Métodos

- DOMNode->appendChild() - Adds new child at the end of the children
- DOMNode->cloneNode() - Clones a node
- DOMNode->hasAttributes() - Checks if node has attributes
- DOMNode->hasChildNodes() - Checks if node has children
- DOMNode->insertBefore() - Adds a new child before a reference node
- DOMNode->isSameNode() - Indicates if two nodes are the same node
- DOMNode->isSupported() - Checks if feature is supported for specified version
- DOMNode->lookupNamespaceURI() - Returns namespace URI of the node based on the prefix
- DOMNode->lookupPrefix() - Returns name space prefix of the node based on namespaceURI
- DOMNode->normalize() - Normalizes the node
- DOMNode->removeChild() - Removes child from list of children
- DOMNode->replaceChild() - Replaces a child

Propiedades

Tabla 9.

Name	Type	Read-only	Description
nodeName	string	yes	Returns the more accurate name for the current node type
nodeValue	string	no	The value of this node, depending on its type.
nodeType	int	yes	Gets the type of the node. One of the predefined XML_xxx_NODE constants
parentNode	DOMNode	yes	The parent of this node.

Name	Type	Read-only	Description
childNodes	DOMNodeList	yes	A **DOMNodeList** that contains all children of this node. If there are no children, this is an empty **DOMNodeList**.
firstChild	DOMNode	yes	The first child of this node. If there is no such node, this returns **NULL**.
lastChild	DOMNode	yes	The last child of this node. If there is no such node, this returns **NULL**.
previousSibling	DOMNode	yes	The node immediately preceding this node. If there is no such node, this returns **NULL**.
nextSibling	DOMNode	yes	The node immediately following this node. If there is no such node, this returns **NULL**.
attributes	DOMNamedNodeMap	yes	A **DOMNamedNodeMap** containing the attributes of this node (if it is a **DOMElement**) or **NULL** otherwise.
ownerDocument	DOMDocument	yes	The **DOMDocument** object associated with this node.
namespaceURI	string	yes	The namespace URI of this node, or **NULL** if it is unspecified.
prefix	string	no	The namespace prefix of this node, or **NULL** if it is unspecified.
localName	string	yes	Returns the local part of the qualified name of this node.
baseURI	string	yes	The absolute base URI of this node or **NULL** if the implementation wasn't able to obtain an absolute URI.
textContent	string	no	This attribute returns the text content of this node and its descendants.

DOMNotation

Extends **DOMNode**

Propiedades

Tabla 10.

Name	Type	Read-only	Description
publicId	string	yes	
systemId	string	yes	

DOMProcessingInstruction

Extends **DOMNode**.

Propiedades

Tabla 11.

Name	Type	Read-only	Description
target	string	yes	
data	string	no	

DOMText

Métodos

- DOMText->isWhitespaceInElementContent() - Indicates whether this text node contains whitespace
- DOMText->splitText() - Breaks the node into two nodes at the specified offset

---

Propiedades

Tabla 12.

Name	Type	Read-only	Description
wholeText	string	yes	

DOMXPath

Constructor

- DOMXPath->__construct() - construct a new DOMXPath object

Métodos

- DOMXPath->registerNamespace() - Registers the namespace with the DOMXpath object

- DOMXPath->evaluate() - Evaluates the given XPath expression and returns a typed result if possible
- DOMXPath->query() - Evaluates the given XPath expression

Propiedades

Tabla 13.

Name	Type	Read-only	Description
document	DOMDocument		

Ejemplos

Many examples in this reference require an XML file. We will use the `book.xml` that contains the following:

Ejemplo 1. chapter.xml

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook XML V4.1.2//EN"
 "http://www.oasis-open.org/docbook/xml/4.1.2/docbookx.dtd" [
]>
<book id="listing">
 <title>My lists</title>
 <chapter id="books">
  <title>My books</title>
  <para>
   <informaltable>
    <tgroup cols="4">
     <thead>
      <row>
       <entry>Title</entry>
       <entry>Author</entry>
       <entry>Language</entry>
       <entry>ISBN</entry>
      </row>
     </thead>
     <tbody>
      <row>
       <entry>The Grapes of Wrath</entry>
       <entry>John Steinbeck</entry>
       <entry>en</entry>
       <entry>0140186409</entry>
      </row>
      <row>
       <entry>The Pearl</entry>
       <entry>John Steinbeck</entry>
       <entry>en</entry>
       <entry>014017737X</entry>
      </row>
      <row>
       <entry>Samarcande</entry>
       <entry>Amine Maalouf</entry>
       <entry>fr</entry>
       <entry>2253051209</entry>
      </row>
      <!-- TODO: I have a lot of remaining books to add.. -->
     </tbody>
    </tgroup>
   </informaltable>
  </para>
 </chapter>
</book>
```

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

Tabla 14. XML constants

Constant	Value	Description
XML_ELEMENT_NODE (integer)	1	Node is a **DOMElement**
XML_ATTRIBUTE_NODE (integer)	2	Node is a **DOMAttr**
XML_TEXT_NODE (integer)	3	Node is a **DOMText**
XML_CDATA_SECTION_NODE (integer)	4	Node is a **DOMCharacterData**
XML_ENTITY_REF_NODE (integer)	5	Node is a **DOMEntityReference**

Constant	Value	Description
XML_ENTITY_NODE (integer)	6	Node is a **DOMEntity**
XML_PI_NODE (integer)	7	Node is a **DOMProcessingInstruction**
XML_COMMENT_NODE (integer)	8	Node is a **DOMComment**
XML_DOCUMENT_NODE (integer)	9	Node is a **DOMDocument**
XML_DOCUMENT_TYPE_NODE (integer)	10	Node is a **DOMDocumentType**
XML_DOCUMENT_FRAG_NODE (integer)	11	Node is a **DOMDocumentFragment**
XML_NOTATION_NODE (integer)	12	Node is a **DOMNotation**
XML_HTML_DOCUMENT_NODE (integer)	13	
XML_DTD_NODE (integer)	14	
XML_ELEMENT_DECL_NODE (integer)	15	
XML_ATTRIBUTE_DECL_NODE (integer)	16	
XML_ENTITY_DECL_NODE (integer)	17	
XML_NAMESPACE_DECL_NODE (integer)	18	
XML_ATTRIBUTE_CDATA (integer)	1	
XML_ATTRIBUTE_ID (integer)	2	
XML_ATTRIBUTE_IDREF (integer)	3	
XML_ATTRIBUTE_IDREFS (integer)	4	
XML_ATTRIBUTE_ENTITY (integer)	5	
XML_ATTRIBUTE_NMTOKEN (integer)	7	
XML_ATTRIBUTE_NMTOKENS (integer)	8	
XML_ATTRIBUTE_ENUMERATION (integer)	9	
XML_ATTRIBUTE_NOTATION (integer)	10	

Tabla 15. DOMException constants

Constant	Value	Description
DOM_INDEX_SIZE_ERR (integer)	1	If index or size is negative, or greater than the allowed value.
DOMSTRING_SIZE_ERR (integer)	2	If the specified range of text does not fit into a **DOMString**.
DOM_HIERARCHY_REQUEST_ERR (integer)	3	If any node is inserted somewhere it doesn't belong
DOM_WRONG_DOCUMENT_ERR (integer)	4	If a node is used in a different document than the one that created it.
DOM_INVALID_CHARACTER_ERR (integer)	5	If an invalid or illegal character is specified, such as in a name.
DOM_NO_DATA_ALLOWED_ERR (integer)	6	If data is specified for a node which does not support data.

Constant	Value	Description
DOM_NO_MODIFICATION_ALLOWED_ERR (integer)	7	If an attempt is made to modify an object where modifications are not allowed.
DOM_NOT_FOUND_ERR (integer)	8	If an attempt is made to reference a node in a context where it does not exist.
DOM_NOT_SUPPORTED_ERR (integer)	9	If the implementation does not support the requested type of object or operation.
DOM_INUSE_ATTRIBUTE_ERR (integer)	10	If an attempt is made to add an attribute that is already in use elsewhere.
DOM_INVALID_STATE_ERR (integer)	11	If an attempt is made to use an object that is not, or is no longer, usable.
DOM_SYNTAX_ERR (integer)	12	If an invalid or illegal string is specified.
DOM_INVALID_MODIFICATION_ERR (integer)	13	If an attempt is made to modify the type of the underlying object.
DOM_NAMESPACE_ERR (integer)	14	If an attempt is made to create or change an object in a way which is incorrect with regard to namespaces.
DOM_INVALID_ACCESS_ERR (integer)	15	If a parameter or an operation is not supported by the underlying object.
DOM_VALIDATION_ERR (integer)	16	If a call to a method such as insertBefore or removeChild would make the Node invalid with respect to "partial validity", this exception would be raised and the operation would not be done.

Tabla de contenidos

DOMDocument->normalize() -- Normalizes the document
DOMDocument->relaxNGValidate() -- Performs relaxNG validation on the document
DOMDocument->relaxNGValidateSource() -- Performs relaxNG validation on the document
DOMDocument->save() -- Dumps the internal XML tree back into a file
DOMDocument->saveHTML() -- Dumps the internal document into a string using HTML formatting
DOMDocument->saveHTMLFile() -- Dumps the internal document into a file using HTML formatting
DOMDocument->saveXML() -- Dumps the internal XML tree back into a string
DOMDocument->schemaValidate() -- Validates a document based on a schema
DOMDocument->schemaValidateSource() -- Validates a document based on a schema
DOMDocument->validate() -- Validates the document based on its DTD
DOMDocument->xinclude() -- Substitutes XIncludes in a DOMDocument Object
DOMElement->getAttribute() -- Returns value of attribute
DOMElement->getAttributeNode() -- Returns attribute node
DOMElement->getAttributeNodeNS() -- Returns attribute node
DOMElement->getAttributeNS() -- Returns value of attribute
DOMElement->getElementsByTagName() -- Gets elements by tagname
DOMElement->getElementsByTagNameNS() -- Get elements by namespaceURI and localName
DOMElement->hasAttribute() -- Checks to see if attribute exists
DOMElement->hasAttributeNS() -- Checks to see if attribute exists
DOMElement->removeAttribute() -- Removes attribute
DOMElement->removeAttributeNode() -- Removes attribute
DOMElement->removeAttributeNS() -- Removes attribute
DOMElement->setAttribute() -- Adds new attribute
DOMElement->setAttributeNode() -- Adds new attribute node to element
DOMElement->setAttributeNodeNS() -- Adds new attribute node to element
DOMElement->setAttributeNS() -- Adds new attribute
DOMImplementation->createDocument() -- Creates a DOMDocument object of the specified type with its document element
DOMImplementation->createDocumentType() -- Creates an empty DOMDocumentType object
DOMImplementation->hasFeature() -- Test if the DOM implementation implements a specific feature
DOMNamedNodeMap->getNamedItem() -- Retrieves a node specified by name
DOMNamedNodeMap->getNamedItemNS() -- Retrieves a node specified by local name and namespace URI
DOMNamedNodeMap->item() -- Retrieves a node specified by index
DOMNode->appendChild() -- Adds new child at the end of the children
DOMNode->cloneNode() -- Clones a node
DOMNode->hasAttributes() -- Checks if node has attributes
DOMNode->hasChildNodes() -- Checks if node has children
DOMNode->insertBefore() -- Adds a new child before a reference node
DOMNode->isSameNode() -- Indicates if two nodes are the same node
DOMNode->isSupported() -- Checks if feature is supported for specified version
DOMNode->lookupNamespaceURI() -- Gets the namespace URI of the node based on the prefix
DOMNode->lookupPrefix() -- Gets the namespace prefix of the node based on the namespace URI
DOMNode->normalize() -- Normalizes the node
DOMNode->removeChild() -- Removes child from list of children
DOMNode->replaceChild() -- Replaces a child
DOMNodelist->item() -- Retrieves a node specified by index
DOMText->isWhitespaceInElementContent() -- Indicates whether this text node contains whitespace
DOMText->splitText() -- Breaks this node into two nodes at the specified offset
DOMXPath->__construct() -- Creates a new DOMXPath object
DOMXPath->evaluate() -- Evaluates the given XPath expression and returns a typed result if

possible.
DOMXPath->query() -- Evaluates the given XPath expression
DOMXPath->registerNamespace() -- Registers the namespace with the DOMXpath object
dom_import_simplexml -- Gets a DOMElement object from a SimpleXMLElement object

DOMAttr->isId()

DOMAttr->isId() -- Checks if attribute is a defined ID

Descripción

class **DOMAttr** {

bool **isId** (void)

}

This function checks if the attribute is a defined ID.

According to the DOM standard this requires a DTD which defines the attribute ID to be of type ID. You need to validate your document with DOMDocument->validate() or *DOMDocument::validateOnParse* before using this function.

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Ejemplos

Ejemplo 1. DOMAttr->isId() Example

```
<?php

$doc = new DomDocument;

// We need to validate our document before refering to the id
$doc->validateOnParse = true;
$doc->Load('book.xml');

// We retrieve the attribute named id of the chapter element
$attr = $doc->getElementsByTagName('chapter')->item(0)->getAttributeNode('id');

var_dump($attr->isId()); // bool(true)

?>
```

DOMCharacterData->appendData()

DOMCharacterData->appendData() -- Append the string to the end of the character data of the node

Descripción

class **DOMCharacterData** {

void **appendData** (string data)

}

Append the string *data* to the end of the character data of the node.

Lista de parámetros

data

The string to append.

Valores retornados

No value is returned.

Ver también

DOMCharacterData->deleteData()
DOMCharacterData->insertData()
DOMCharacterData->replaceData()
DOMCharacterData->substringData
()

DOMCharacterData->deleteData()

DOMCharacterData->deleteData() -- Remove a range of characters from the node

Descripción

class **DOMCharacterData** {

void **deleteData** (int offset, int count)

}

Deletes *count* characters starting from position *offset*.

Lista de parámetros

offset

The offset from which to start removing.

count

The number of characters to delete. If the sum of *offset* and *count* exceeds the length, then all characters to the end of the data are deleted.

Valores retornados

No value is returned.

Exceptions

`DOM_INDEX_SIZE_ERR`

Raised if *offset* is negative or greater than the number of 16-bit units in data, or if *count* is negative.

Ver también

DOMCharacterData->appendData()
DOMCharacterData->insertData()
DOMCharacterData->replaceData()
DOMCharacterData->substringData()

DOMCharacterData->insertData()

DOMCharacterData->insertData() -- Insert a string at the specified 16-bit unit offset

Descripción

class **DOMCharacterData** {

void **insertData** (int offset, string data)

}

Inserts string *data* at position *offset*.

Lista de parámetros

offset

The character offset at which to insert.

data

The string to insert.

Valores retornados

No value is returned.

Exceptions

`DOM_INDEX_SIZE_ERR`

Raised if *offset* is negative or greater than the number of 16-bit units in data.

Ver también

DOMCharacterData->appendData()
DOMCharacterData->deleteData()
DOMCharacterData->replaceData()
DOMCharacterData->substringData()

DOMCharacterData->replaceData()

DOMCharacterData->replaceData() -- Replace a substring within the DOMCharacterData node

Descripción

class **DOMCharacterData** {

void **replaceData** (int offset, int count, string data)

}

Replace *count* characters starting from position *offset* with *data*.

Lista de parámetros

offset

The offset from which to start replacing.

count

The number of characters to replace. If the sum of *offset* and *count* exceeds the length, then all characters to the end of the data are replaced.

data

The string with which the range must be replaced.

Valores retornados

No value is returned.

Exceptions

DOM_INDEX_SIZE_ERR

Raised if *offset* is negative or greater than the number of 16-bit units in data, or if *count* is negative.

Ver también

DOMCharacterData->appendData()
DOMCharacterData->deleteData()
DOMCharacterData->insertData()
DOMCharacterData->substringData()

DOMCharacterData->substringData()

DOMCharacterData->substringData() -- Extracts a range of data from the node

Descripción

class **DOMCharacterData** {

string **substringData** (int offset, int count)

}

Returns the specified substring.

Lista de parámetros

offset

Start offset of substring to extract.

count

The number of characters to extract.

Valores retornados

The specified substring. If the sum of *offset* and *count* exceeds the length, then all 16-bit units to the end of the data are returned.

Exceptions

DOM_INDEX_SIZE_ERR

Raised if *offset* is negative or greater than the number of 16-bit units in data, or if *count* is

negative.

Ver también

DOMCharacterData->appendData()
DOMCharacterData->deleteData()
DOMCharacterData->insertData()
DOMCharacterData->replaceData()

DOMDocument->__construct()

DOMDocument->__construct() -- Creates a new DOMDocument object

Descripción

class **DOMDocument** {

__construct ([string version [, string encoding]])

}

Creates a new **DOMDocument** object.

Lista de parámetros

version

The version number of the document as part of the XML declaration.

encoding

The encoding of the document as part of the XML declaration.

Ejemplos

Ejemplo 1. Creating a new DOMDocument

```
<?php

$dom = new DOMDocument('1.0', 'iso-8859-1');

echo $dom->saveXML(); /* <?xml version="1.0" encoding="iso-8859-1"?> */

?>
```

Ver también

DOMImplementation->createDocument
()

DOMDocument->createAttribute()

DOMDocument->createAttribute() -- Create new attribute

Descripción

class **DOMDocument** {

DOMAttr **createAttribute** (string name)

}

This function creates a new instance of class **DOMAttr**. Este nodo no sera mostrado en el documento a no ser que sea introducido por ejemplo con DOMNode->appendChild().

Lista de parámetros

name

The name of the attribute.

Valores retornados

The new **DOMAttr** or **`FALSE`** if an error occured.

Exceptions

`DOM_INVALID_CHARACTER_ERR`

Raised if *name* contains an invalid character.

Ver también

DOMNode->appendChild()
DOMDocument->createAttributeNS()
DOMDocument->createCDATASection()
DOMDocument->createComment()
DOMDocument->createDocumentFragment()
DOMDocument->createElement()
DOMDocument->createElementNS()
DOMDocument->createEntityReference()
DOMDocument->createProcessingInstruction
()
DOMDocument->createTextNode()

DOMDocument->createAttributeNS()

DOMDocument->createAttributeNS() -- Create new attribute node with an associated namespace

Descripción

class **DOMDocument** {

DOMAttr **createAttributeNS** (string namespaceURI, string qualifiedName)

}

This function creates a new instance of class **DOMAttr**. Este nodo no sera mostrado en el documento a no ser que sea introducido por ejemplo con DOMNode->appendChild().

Lista de parámetros

namespaceURI

The URI of the namespace.

qualifiedName

The tag name and prefix of the attribute, as *prefix:tagname*.

Valores retornados

The new **DOMAttr** or **`FALSE`** if an error occured.

Exceptions

`DOM_INVALID_CHARACTER_ERR`

Raised if *qualifiedName* contains an invalid character.

`DOM_NAMESPACE_ERR`

Raised if *qualifiedName* is a malformed qualified name, or if *qualifiedName* has a prefix and *namespaceURI* is **`NULL`**.

Ver también

DOMNode->appendChild()
DOMDocument->createAttribute()
DOMDocument->createCDATASection()
DOMDocument->createComment()
DOMDocument->createDocumentFragment()
DOMDocument->createElement()
DOMDocument->createElementNS()
DOMDocument->createEntityReference()
DOMDocument->createProcessingInstruction
()
DOMDocument->createTextNode()

DOMDocument->createCDATASection()

DOMDocument->createCDATASection() -- Create new cdata node

Descripción

class **DOMDocument** {

DOMCDATASection **createCDATASection** (string data)

}

This function creates a new instance of class **DOMCDATASection**. Este nodo no sera mostrado en el documento a no ser que sea introducido por ejemplo con DOMNode->appendChild().

Lista de parámetros

data

The content of the cdata.

Valores retornados

The new **DOMCDATASection** or **`FALSE`** if an error occured.

Ver también

DOMNode->appendChild()
DOMDocument->createAttribute()
DOMDocument->createAttributeNS()
DOMDocument->createComment()
DOMDocument->createDocumentFragment()
DOMDocument->createElement()
DOMDocument->createElementNS()
DOMDocument->createEntityReference()
DOMDocument->createProcessingInstruction()
DOMDocument->createTextNode()

DOMDocument->createComment()

DOMDocument->createComment() -- Create new comment node

Descripción

class **DOMDocument** {

DOMComment **createComment** (string data)

}

This function creates a new instance of class **DOMComment**. Este nodo no sera mostrado en el documento a no ser que sea introducido por ejemplo con DOMNode->appendChild().

Lista de parámetros

data

The content of the comment.

Valores retornados

The new **DOMComment** or **`FALSE`** if an error occured.

Ver también

DOMNode->appendChild()
DOMDocument->createAttribute()
DOMDocument->createAttributeNS()
DOMDocument->createCDATASection()
DOMDocument->createDocumentFragment()
DOMDocument->createElement()
DOMDocument->createElementNS()
DOMDocument->createEntityReference()
DOMDocument->createProcessingInstruction()
DOMDocument->createTextNode()

DOMDocument->createDocumentFragment()

DOMDocument->createDocumentFragment() -- Create new document fragment

Descripción

class **DOMDocument** {

DOMDocumentFragment **createDocumentFragment** (void)

}

This function creates a new instance of class **DOMDocumentFragment**. Este nodo no sera mostrado en el documento a no ser que sea introducido por ejemplo con DOMNode->appendChild().

Valores retornados

The new **DOMDocumentFragment** or **`FALSE`** if an error occured.

Ver también

DOMNode->appendChild()
DOMDocument->createAttribute()
DOMDocument->createAttributeNS()
DOMDocument->createCDATASection()
DOMDocument->createComment()
DOMDocument->createElement()
DOMDocument->createElementNS()
DOMDocument->createEntityReference()
DOMDocument->createProcessingInstruction
()
DOMDocument->createTextNode()

DOMDocument->createElement()

DOMDocument->createElement() -- Create new element node

Descripción

class **DOMDocument** {

DOMElement **createElement** (string name [, string value])

}

This function creates a new instance of class **DOMElement**. Este nodo no sera mostrado en el documento a no ser que sea introducido por ejemplo con DOMNode->appendChild().

Lista de parámetros

name

The tag name of the element.

value

The value of the element. By default, an empty element will be created. You can also set the value later with *DOMElement->nodeValue*.

Valores retornados

Returns a new instance of class **DOMElement** or **`FALSE`** if an error occured.

Exceptions

`DOM_INVALID_CHARACTER_ERR`

Raised if *name* contains an invalid character.

Ejemplos

Ejemplo 1. Creating a new element and inserting it as root

```
<?php

$dom = new DOMDocument('1.0', 'iso-8859-1');

$element = $dom->createElement('test', 'This is the root element!');

// We insert the new element as root (child of the document)
$dom->appendChild($element);

echo $dom->saveXML();
?>
```

El resultado del ejemplo seria:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<test>This is the root element!</test>
```

Ver también

DOMNode->appendChild()
DOMDocument->createAttribute()
DOMDocument->createAttributeNS()
DOMDocument->createCDATASection()
DOMDocument->createComment()
DOMDocument->createDocumentFragment()
DOMDocument->createElementNS()
DOMDocument->createEntityReference()
DOMDocument->createProcessingInstruction()
DOMDocument->createTextNode()

DOMDocument->createElementNS()

DOMDocument->createElementNS() -- Create new element node with an associated namespace

Descripción

class **DOMDocument** {

DOMElement **createElementNS** (string namespaceURI, string qualifiedName [, string value])

}

This function creates a new element node with an associated namespace. Este nodo no sera mostrado en el documento a no ser que sea introducido por ejemplo con DOMNode->appendChild().

Lista de parámetros

namespaceURI

The URI of the namespace.

qualifiedName

The qualified name of the element, as *prefix:tagname*.

value

The value of the element. By default, an empty element will be created. You can also set the value later with *DOMElement->nodeValue*.

Valores retornados

The new **DOMElement** or **FALSE** if an error occured.

Exceptions

DOM_INVALID_CHARACTER_ERR

Raised if *qualifiedName* contains an invalid character.

DOM_NAMESPACE_ERR

Raised if *qualifiedName* is a maformed qualified name.

Ejemplos

Ejemplo 1. Creating a new element and inserting it as root

```
<?php

$dom = new DOMDocument('1.0', 'iso-8859-1');

$element = $dom->createElementNS('http://www.example.com/XFoo', 'xfoo:test', 'This is t

// We insert the new element as root (child of the document)
$dom->appendChild($element);

echo $dom->saveXML();
?>
```

El resultado del ejemplo seria:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xfoo:test xmlns:xfoo="http://www.example.com/XFoo">This is the root element!</xfoo:tes
```

Ver también

DOMNode->appendChild()
DOMDocument->createAttribute()
DOMDocument->createAttributeNS()
DOMDocument->createCDATASection()
DOMDocument->createComment()
DOMDocument->createDocumentFragment()
DOMDocument->createElement()

DOMDocument->createEntityReference()
DOMDocument->createProcessingInstruction()
DOMDocument->createTextNode()

DOMDocument->createEntityReference()

DOMDocument->createEntityReference() -- Create new entity reference node

Descripción

class **DOMDocument** {

DOMEntityReference **createEntityReference** (string name)

}

This function creates a new instance of class **DOMEntityReference**. Este nodo no sera mostrado en el documento a no ser que sea introducido por ejemplo con DOMNode->appendChild().

Lista de parámetros

name

The content of the entity reference.

Valores retornados

The new **DOMEntityReference** or **`FALSE`** if an error occured.

Exceptions

`DOM_INVALID_CHARACTER_ERR`

Raised if *name* contains an invalid character.

Ver también

DOMNode->appendChild()
DOMDocument->createAttribute()
DOMDocument->createAttributeNS()
DOMDocument->createCDATASection()
DOMDocument->createComment()
DOMDocument->createDocumentFragment()
DOMDocument->createElement()
DOMDocument->createElementNS()
DOMDocument->createProcessingInstruction()

DOMDocument->createTextNode()

DOMDocument->createProcessingInstruction()

DOMDocument->createProcessingInstruction() -- Creates new PI node

Descripción

class **DOMDocument** {

DOMProcessingInstruction **createProcessingInstruction** (string target [, string data])

}

This function creates a new instance of class **DOMProcessingInstruction**. Este nodo no sera mostrado en el documento a no ser que sea introducido por ejemplo con DOMNode->appendChild().

Lista de parámetros

target

The target of the processing instruction.

data

The content of the processing instruction.

Valores retornados

The new **DOMProcessingInstruction** or **`FALSE`** if an error occured.

Exceptions

`DOM_INVALID_CHARACTER_ERR`

Raised if *target* contains an invalid character.

Ver también

DOMNode->appendChild()
DOMDocument->createAttribute()
DOMDocument->createAttributeNS()
DOMDocument->createCDATASection()
DOMDocument->createComment()
DOMDocument->createDocumentFragment()

DOMDocument->createElement()
DOMDocument->createElementNS()
DOMDocument->createEntityReference()
DOMDocument->createTextNode()

DOMDocument->createTextNode()

DOMDocument->createTextNode() -- Create new text node

Descripción

class **DOMDocument** {

DOMText **createTextNode** (string content)

}

This function creates a new instance of class **DOMText**. Este nodo no sera mostrado en el documento a no ser que sea introducido por ejemplo con DOMNode->appendChild().

Lista de parámetros

content

The content of the text.

Valores retornados

The new **DOMText** or **`FALSE`** if an error occured.

Ver también

DOMNode->appendChild()
DOMDocument->createAttribute()
DOMDocument->createAttributeNS()
DOMDocument->createCDATASection()
DOMDocument->createComment()
DOMDocument->createDocumentFragment()
DOMDocument->createElement()
DOMDocument->createElementNS()
DOMDocument->createEntityReference()
DOMDocument->createProcessingInstruction
()

DOMDocument->getElementById()

DOMDocument->getElementById() -- Searches for an element with a certain id

Descripción

class **DOMDocument** {

DOMElement **getElementById** (string elementId)

}

This function is similar to DOMDocument->getElementsByTagName() but searches for an element with a given id.

According to the DOM standard this requires a DTD which defines the attribute ID to be of type ID. You need to validate your document with DOMDocument->validate() or *DOMDocument->validateOnParse* before using this function.

Lista de parámetros

elementId

The unique id value for an element.

Valores retornados

Returns the **DOMElement** or **`NULL`** if the element is not found.

Ejemplos

Ejemplo 1. DOMDocument->getElementById() Example

```
<?php

$doc = new DomDocument;

// We need to validate our document before refering to the id
$doc->validateOnParse = true;
$doc->Load('book.xml');

echo "The element whose id is books is: " . $doc->getElementById('books')->tagName . "\

?>
```

El resultado del ejemplo seria:

```
The element whose id is books is: chapter
```

Ver también

DOMDocument->getElementsByTagName
()

DOMDocument->getElementsByTagName()

DOMDocument->getElementsByTagName() -- Searches for all elements with given tag name

Descripción

class **DOMDocument** {

DOMNodeList **getElementsByTagName** (string name)

}

This function returns a new instance of class **DOMNodeList** containing the elements with a given tag name.

Lista de parámetros

name

The name of the tag to match on. The special value * matches all tags.

Valores retornados

A new **DOMNodeList** object containing all the matched elements.

Ver también

DOMDocument->getElementsByTagNameNS
()

DOMDocument->getElementsByTagNameNS ()

DOMDocument->getElementsByTagNameNS() -- Searches for all elements with given tag name in specified namespace

Descripción

class **DOMDocument** {

DOMNodeList **getElementsByTagNameNS** (string namespaceURI, string localName)

}

Returns a **DOMNodeList** of all elements with a given local name and a namespace URI.

Lista de parámetros

namespaceURI

The namespace URI of the elements to match on. The special value * matches all namespaces.

localName

The local name of the elements to match on. The special value * matches all local names.

Valores retornados

A new **DOMNodeList** object containing all the matched elements.

Ejemplos

Ejemplo 1. Get all the XInclude elements

```
<?php

$xml = <<<EOD
<?xml version="1.0" ?>
<chapter xmlns:xi="http://www.w3.org/2001/XInclude">
<title>Books of the other guy..</title>
<para>
 <xi:include href="book.xml">
  <xi:fallback>
   <error>xinclude: book.xml not found</error>
  </xi:fallback>
 </xi:include>
 <include>
  This is another namespace
 </include>
</para>
</chapter>
EOD;
$dom = new DOMDocument;

// load the XML string defined above
$dom->loadXML($xml);

foreach ($dom->getElementsByTagNameNS('http://www.w3.org/2001/XInclude', '*') as $eleme
    echo 'local name: ', $element->localName, ', prefix: ', $element->prefix, "\n";
}
?>
```

El resultado del ejemplo seria:

```
local name: include, prefix: xi
local name: fallback, prefix: xi
```

Ver también

DOMDocument->getElementsByTagName
()

DOMDocument->importNode()

DOMDocument->importNode() -- Import node into current document

Descripción

class **DOMDocument** {

DOMNode **importNode** (DOMNode importedNode [, bool deep])

}

This function returns a copy of the node to import and associates it with the current document.

Lista de parámetros

importedNode

The node to import.

deep

If set to **TRUE**, this method will recursively import the subtree under the *importedNode*.

Valores retornados

The copied node.

Exceptions

DOMException is thrown if node cannot be imported.

DOMDocument->load()

DOMDocument->load() -- Load XML from a file

Descripción

class **DOMDocument** {

bool **load** (string filename)

}

Loads an XML document from a file.

This method may also be called statically to load and create a **DOMDocument** object. The static invocation may be used when no **DOMDocument** properties need to be set prior to loading.

Lista de parámetros

filename

The path to the XML document.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplos

Ejemplo 1. Creating a Document

```
<?php
$doc = DOMDocument::load('book.xml');
echo $doc->saveXML();

$doc = new DOMDocument();
$doc->load('book.xml');
echo $doc->saveXML();
?>
```

Ver también

DOMDocument->loadXML
()
DOMDocument->save()
DOMDocument->saveXML
()

DOMDocument->loadHTML()

DOMDocument->loadHTML() -- Load HTML from a string

Descripción

class **DOMDocument** {

bool **loadHTML** (string source)

}

The function parses the HTML contained in the string *source*. Unlike loading XML, HTML does not have to be well-formed to load. This function may also be called statically to load and create a **DOMDocument** object. The static invocation may be used when no **DOMDocument** properties need to be set prior to loading.

Lista de parámetros

source

The HTML string.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplos

Ejemplo 1. Creating a Document

```
<?php
$doc = DOMDocument::loadHTML("<html><body>Test<br></body></html>");
echo $doc->saveHTML();

$doc = new DOMDocument();
$doc->loadHTML("<html><body>Test<br></body></html>");
echo $doc->saveHTML();
?>
```

Ver también

DOMDocument->loadHTMLFile
()
DOMDocument->saveHTML()
DOMDocument->saveHTMLFile
()

DOMDocument->loadHTMLFile()

DOMDocument->loadHTMLFile() -- Load HTML from a file

Descripción

class **DOMDocument** {

bool **loadHTMLFile** (string filename)

}

The function parses the HTML document in the file named *filename*. Unlike loading XML, HTML does not have to be well-formed to load.

This function may also be called statically to load and create a **DOMDocument** object. The static invocation may be used when no **DOMDocument** properties need to be set prior to loading.

Lista de parámetros

filename

The path to the HTML file.

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Ejemplos

Ejemplo 1. Creating a Document

```
<?php
$doc = DOMDocument::loadHTMLFile("filename.html");
print $doc->saveHTML();

$doc = new DOMDocument();
$doc->loadHTMLFile("filename.html");
print $doc->saveHTML();
?>
```

Ver también

DOMDocument->loadHTML()
DOMDocument->saveHTML()
DOMDocument->saveHTMLFile
()

DOMDocument->loadXML()

DOMDocument->loadXML() -- Load XML from a string

Descripción

class **DOMDocument** {

bool **loadXML** (string source)

}

Loads an XML document from a string.

This method may also be called statically to load and create a **DOMDocument** object. The static invocation may be used when no **DOMDocument** properties need to be set prior to loading.

Lista de parámetros

source

The string containing the XML.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplos

Ejemplo 1. Creating a Document

```
<?php
$doc = DOMDocument::loadXML('<root><node/></root>');
echo $doc->saveXML();

$doc = new DOMDocument();
$doc->loadXML('<root><node/></root>');
echo $doc->saveXML();
?>
```

Ver también

DOMDocument->load()
DOMDocument->save()
DOMDocument->saveXML
()

DOMDocument->normalize()

DOMDocument->normalize() -- Normalizes the document

Descripción

class **DOMDocument** {

void **normalize** (void)

}

Normalizes the document.

Valores retornados

No value is returned.

Ver también

DOMNode->normalize
()

DOMDocument->relaxNGValidate()

DOMDocument->relaxNGValidate() -- Performs relaxNG validation on the document

Descripción

class **DOMDocument** {

bool **relaxNGValidate** (string filename)

}

Performs relaxNG validation on the document based on the given RNG schema.

Lista de parámetros

filename

The RNG file.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ver también

DOMDocument->relaxNGValidateSource
()
DOMDocument->schemaValidate()
DOMDocument->schemaValidateSource
()
DOMDocument->validate()

DOMDocument->relaxNGValidateSource()

DOMDocument->relaxNGValidateSource() -- Performs relaxNG validation on the document

Descripción

class **DOMDocument** {

bool **relaxNGValidateSource** (string source)

}

Performs relaxNG validation on the document based on the given RNG source.

Lista de parámetros

source

A string containing the RNG schema.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ver también

DOMDocument->relaxNGValidate()
DOMDocument->schemaValidate()

DOMDocument->schemaValidateSource
()
DOMDocument->validate()

DOMDocument->save()

DOMDocument->save() -- Dumps the internal XML tree back into a file

Descripción

class **DOMDocument** {

mixed **save** (string filename)

}

Creates an XML document from the DOM representation. This function is usually called after building a new dom document from scratch as in the example below.

Lista de parámetros

filename

The path to the saved XML document.

Valores retornados

Returns the number of bytes written or **FALSE** if an error occurred.

Ejemplos

Ejemplo 1. Saving a DOM tree into a file

```
<?php

$doc = new DOMDocument('1.0');
// we want a nice output
$doc->formatOutput = true;

$root = $doc->createElement('book');
$root = $doc->appendChild($root);

$title = $doc->createElement('title');
$title = $root->appendChild($title);

$text = $doc->createTextNode('This is the title');
$text = $title->appendChild($text);

echo 'Wrote: ' . $doc->save("/tmp/test.xml") . ' bytes'; // Wrote: 72 bytes

?>
```

Ver también

DOMDocument->saveXML
()
DOMDocument->load()
DOMDocument->loadXML
()

DOMDocument->saveHTML()

DOMDocument->saveHTML() -- Dumps the internal document into a string using HTML formatting

Descripción

class **DOMDocument** {

string **saveHTML** (void)

}

Creates an HTML document from the DOM representation. This function is usually called after building a new dom document from scratch as in the example below.

Valores retornados

Returns the HTML, or **`FALSE`** if an error occurred.

Ejemplos

Ejemplo 1. Saving a HTML tree into a string

```
<?php

$doc = new DOMDocument('1.0');
// we want a nice output
$doc->formatOutput = true;

$root = $doc->createElement('html');
$root = $doc->appendChild($root);

$head = $doc->createElement('head');
$head = $root->appendChild($head);

$title = $doc->createElement('title');
$title = $head->appendChild($title);

$text = $doc->createTextNode('This is the title');
$text = $title->appendChild($text);

echo $doc->saveHTML();

?>
```

Ver también

DOMDocument->saveHTMLFile
()
DOMDocument->loadHTML()
DOMDocument->loadHTMLFile
()

DOMDocument->saveHTMLFile()

DOMDocument->saveHTMLFile() -- Dumps the internal document into a file using HTML formatting

Descripción

class **DOMDocument** {

int **saveHTMLFile** (string filename)

}

Creates an HTML document from the DOM representation. This function is usually called after building a new dom document from scratch as in the example below.

Lista de parámetros

filename

The path to the saved HTML document.

Valores retornados

Returns the number of bytes written or **`FALSE`** if an error occurred.

Ejemplos

Ejemplo 1. Saving a HTML tree into a file

```php
<?php

$doc = new DOMDocument('1.0');
// we want a nice output
$doc->formatOutput = true;

$root = $doc->createElement('html');
$root = $doc->appendChild($root);

$head = $doc->createElement('head');
$head = $root->appendChild($head);

$title = $doc->createElement('title');
$title = $head->appendChild($title);

$text = $doc->createTextNode('This is the title');
$text = $title->appendChild($text);

echo 'Wrote: ' . $doc->saveHTMLFile("/tmp/test.html") . ' bytes'; // Wrote: 129 bytes

?>
```

Ver también

DOMDocument->saveHTML()
DOMDocument->loadHTML()
DOMDocument->loadHTMLFile
()

DOMDocument->saveXML()

DOMDocument->saveXML() -- Dumps the internal XML tree back into a string

Descripción

class **DOMDocument** {

string **saveXML** ([DOMNode node])

}

Creates an XML document from the DOM representation. This function is usually called after building a new dom document from scratch as in the example below.

Lista de parámetros

node

Use this parameter to output only a specific node without XML declaration rather than the entire document.

Valores retornados

Returns the XML, or **FALSE** if an error occurred.

Exceptions

DOM_WRONG_DOCUMENT_ERR

Raised if *node* is from another document.

Ejemplos

Ejemplo 1. Saving a DOM tree into a string

```
<?php

$doc = new DOMDocument('1.0');
// we want a nice output
$doc->formatOutput = true;

$root = $doc->createElement('book');
$root = $doc->appendChild($root);

$title = $doc->createElement('title');
$title = $root->appendChild($title);

$text = $doc->createTextNode('This is the title');
$text = $title->appendChild($text);

echo "Retrieving all the document:\n";
echo $doc->saveXML() . "\n";

echo "Retrieving only the title part:\n";
echo $doc->saveXML($title);

?>
```

El resultado del ejemplo seria:

```
Retrieving all the document:
<?xml version="1.0"?>
<book>
  <title>This is the title</title>
</book>

Retrieving only the title part:
<title>This is the title</title>
```

Ver también

DOMDocument->save()
DOMDocument->load()
DOMDocument->loadXML
()

DOMDocument->schemaValidate()

DOMDocument->schemaValidate() -- Validates a document based on a schema

Descripción

class **DOMDocument** {

bool **schemaValidate** (string filename)

}

Validates a document based on a the given schema file.

Lista de parámetros

filename

The path to the schema.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ver también

DOMDocument->schemaValidateSource
()
DOMDocument->relaxNGValidate()
DOMDocument->relaxNGValidateSource
()
DOMDocument->validate()

DOMDocument->schemaValidateSource()

DOMDocument->schemaValidateSource() -- Validates a document based on a schema

Descripción

class **DOMDocument** {

bool **schemaValidateSource** (string source)

}

Validates a document based on a schema defined in the given string.

Lista de parámetros

source

A string containing the schema.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ver también

DOMDocument->schemaValidate()
DOMDocument->relaxNGValidate()
DOMDocument->relaxNGValidateSource()
DOMDocument->validate()

DOMDocument->validate()

DOMDocument->validate() -- Validates the document based on its DTD

Descripción

class **DOMDocument** {

bool **validate** (void)

}

Validates the document based on its DTD.

You can also use the *validateOnParse* property of **DOMDocument** to make a DTD validation.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo. If the document have no DTD attached, this method will return **FALSE**.

Ejemplos

Ejemplo 1. Example of DTD validation

```
<?php
$dom = new DOMDocument;
$dom->Load('book.xml');
if ($dom->validate()) {
    echo "This document is valid!\n";
}
?>
```

You can also validate your XML file while loading it:

```
<?php
$dom = new DOMDocument;
$dom->validateOnParse = true;
$dom->Load('book.xml');
?>
```

Ver también

DOMDocument->schemaValidate()
DOMDocument->schemaValidateSource()

DOMDocument->relaxNGValidate()
DOMDocument->relaxNGValidateSource
()

DOMDocument->xinclude()

DOMDocument->xinclude() -- Substitutes XIncludes in a DOMDocument Object

Descripción

class **DOMDocument** {

int **xinclude** ([int options])

}

This method substitutes XIncludes in a DOMDocument object.

> **Nota:** Due to libxml2 automatically resolving entities, this method will produce unexpected results if the included XML file have an attached DTD.

Lista de parámetros

options

> libxml parameters. Available since PHP 5.1.0 and Libxml 2.6.7.

Valores retornados

Returns the number of XIncludes in the document.

Ejemplos

Ejemplo 1. DOMDocument->xinclude() example

```
<?php

$xml = <<<EOD
<?xml version="1.0" ?>
<chapter xmlns:xi="http://www.w3.org/2001/XInclude">
 <title>Books of the other guy..</title>
 <para>
  <xi:include href="book.xml">
   <xi:fallback>
    <error>xinclude: book.xml not found</error>
   </xi:fallback>
  </xi:include>
 </para>
</chapter>
EOD;

$dom = new DOMDocument;

// let's have a nice output
$dom->preserveWhiteSpace = false;
$dom->formatOutput = true;

// load the XML string defined above
$dom->loadXML($xml);

// substitute xincludes
$dom->xinclude();

echo $dom->saveXML();

?>
```

El resultado del ejemplo seria algo similar a:

```
<?xml version="1.0"?>
<chapter xmlns:xi="http://www.w3.org/2001/XInclude">
  <title>Books of the other guy..</title>
  <para>
    <row xml:base="/home/didou/book.xml">
       <entry>The Grapes of Wrath</entry>
       <entry>John Steinbeck</entry>
       <entry>en</entry>
       <entry>0140186409</entry>
      </row>
    <row xml:base="/home/didou/book.xml">
       <entry>The Pearl</entry>
       <entry>John Steinbeck</entry>
       <entry>en</entry>
       <entry>014017737X</entry>
      </row>
    <row xml:base="/home/didou/book.xml">
       <entry>Samarcande</entry>
       <entry>Amine Maalouf</entry>
       <entry>fr</entry>
       <entry>2253051209</entry>
      </row>
  </para>
</chapter>
```

DOMElement->getAttribute()

DOMElement->getAttribute() -- Returns value of attribute

Descripción

class **DOMElement** {

string **getAttribute** (string name)

}

Gets the value of the attribute with name *name* for the current node.

Lista de parámetros

name

The name of the attribute.

Valores retornados

The value of the attribute, or an empty string if no attribute with the given *name* is found.

Ver también

DOMElement->hasAttribute()
DOMElement->setAttribute()
DOMElement->removeAttribute()

DOMElement->getAttributeNode()

DOMElement->getAttributeNode() -- Returns attribute node

Descripción

class **DOMElement** {

DOMAttr **getAttributeNode** (string name)

}

Returns the attribute node with name *name* for the current element.

Lista de parámetros

name

The name of the attribute.

Valores retornados

The attribute node.

Ver también

DOMElement->hasAttribute()
DOMElement->setAttributeNode()
DOMElement->removeAttributeNode
()

DOMElement->getAttributeNodeNS()

DOMElement->getAttributeNodeNS() -- Returns attribute node

Descripción

class **DOMElement** {

DOMAttr **getAttributeNodeNS** (string namespaceURI, string localName)

}

Returns the attribute node in namespace *namespaceURI* with local name *localName* for the current node.

Lista de parámetros

namespaceURI

The namespace URI.

localName

The local name.

Valores retornados

The attribute node.

Ver también

DOMElement->hasAttributeNS()
DOMElement->setAttributeNodeNS()
DOMElement->removeAttributeNode
()

DOMElement->getAttributeNS()

DOMElement->getAttributeNS() -- Returns value of attribute

Descripción

class **DOMElement** {

string **getAttributeNS** (string namespaceURI, string localName)

}

Gets the value of the attribute in namespace *namespaceURI* with local name *localName* for the current node.

Lista de parámetros

namespaceURI

The namespace URI.

localName

The local name.

Valores retornados

The value of the attribute, or an empty string if no attribute with the given *localName* and *namespaceURI* is found.

Ver también

DOMElement->hasAttributeNS()
DOMElement->setAttributeNS()
DOMElement->removeAttributeNS()

DOMElement->getElementsByTagName()

DOMElement->getElementsByTagName() -- Gets elements by tagname

Descripción

class **DOMElement** {

DOMNodeList **getElementsByTagName** (string name)

}

This function returns a new instance of the class **DOMNodeList** of all descendant elements with a given tag *name*, in the order in which they are encountered in a preorder traversal of this element tree.

Lista de parámetros

name

The tag name. Use * to return all elements within the element tree.

Valores retornados

This function returns a new instance of the class **DOMNodeList** of all matched elements.

Ver también

DOMElement->getElementsByTagNameNS
()

DOMElement->getElementsByTagNameNS()

DOMElement->getElementsByTagNameNS() -- Get elements by namespaceURI and localName

Descripción

class **DOMElement** {

DOMNodeList **getElementsByTagNameNS** (string namespaceURI, string localName)

}

This function fetch all the descendant elements with a given *localName* and *namespaceURI*.

Lista de parámetros

namespaceURI

The namespace URI.

localName

The local name. Use * to return all elements within the element tree.

Valores retornados

This function returns a new instance of the class **DOMNodeList** of all matched elements in the order in which they are encountered in a preorder traversal of this element tree.

Ver también

DOMElement-
>getElementsByTagName()

DOMElement->hasAttribute()

DOMElement->hasAttribute() -- Checks to see if attribute exists

Descripción

class **DOMElement** {

bool **hasAttribute** (string name)

}

Indicates whether attribute named *name* exists as a member of the element.

Lista de parámetros

name

The attribute name.

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Ver también

DOMElement->hasAttributeNS
()
DOMElement->getAttribute()
DOMElement->setAttribute()
DOMElement->removeAttribute
()

DOMElement->hasAttributeNS()

DOMElement->hasAttributeNS() -- Checks to see if attribute exists

Descripción

class **DOMElement** {

bool **hasAttributeNS** (string namespaceURI, string localName)

}

Indicates whether attribute in namespace *namespaceURI* named *localName* exists as a member of the element.

Lista de parámetros

namespaceURI

The namespace URI.

localName

The local name.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ver también

DOMElement->hasAttribute()
DOMElement->getAttributeNS()
DOMElement->setAttributeNS()
DOMElement->removeAttributeNS()

DOMElement->removeAttribute()

DOMElement->removeAttribute() -- Removes attribute

Descripción

class **DOMElement** {

bool **removeAttribute** (string name)

}

Removes attribute named *name* from the element.

Lista de parámetros

name

The name of the attribute.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Exceptions

DOM_NO_MODIFICATION_ALLOWED_ERR

Raised if the node is readonly.

Ver también

DOMElement->hasAttribute
()
DOMElement->getAttribute
()
DOMElement->setAttribute
()

DOMElement->removeAttributeNode()

DOMElement->removeAttributeNode() -- Removes attribute

Descripción

class **DOMElement** {

bool **removeAttributeNode** (DOMAttr oldnode)

}

Removes attribute *oldnode* from the element.

Lista de parámetros

oldnode

The attribute node.

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Exceptions

`DOM_NO_MODIFICATION_ALLOWED_ERR`

Raised if the node is readonly.

`DOM_NOT_FOUND_ERROR`

Raised if *oldnode* is not an attribute of the element.

Ver también

DOMElement->hasAttribute()

DOMElement->getAttributeNode
()
DOMElement->setAttributeNode
()

DOMElement->removeAttributeNS()

DOMElement->removeAttributeNS() -- Removes attribute

Descripción

class **DOMElement** {

bool **removeAttributeNS** (string namespaceURI, string localName)

}

Removes attribute is namespace *namespaceURI* named *localName* from the element.

Lista de parámetros

namespaceURI

The namespace URI.

localName

The local name.

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Exceptions

`DOM_NO_MODIFICATION_ALLOWED_ERR`

Raised if the node is readonly.

Ver también

DOMElement->hasAttributeNS
()
DOMElement->getAttributeNS
()
DOMElement->setAttributeNS
()

DOMElement->setAttribute()

DOMElement->setAttribute() -- Adds new attribute

Descripción

class **DOMElement** {

bool **setAttribute** (string name, string value)

}

Sets an attribute with name *name* to the given value. If the attribute does not exist, it will be created.

Lista de parámetros

name

The name of the attribute.

value

The value of the attribute.

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Exceptions

`DOM_NO_MODIFICATION_ALLOWED_ERR`

Raised if the node is readonly.

Ejemplos

Ejemplo 1. Setting an attribute

```
<?php
$doc = new DOMDocument("1.0");
$node = $doc->createElement("para");
$newnode = $doc->appendChild($node);
$newnode->setAttribute("align", "left");
?>
```

Ver también

DOMElement->hasAttribute()
DOMElement->getAttribute()
DOMElement->removeAttribute()

DOMElement->setAttributeNode()

DOMElement->setAttributeNode() -- Adds new attribute node to element

Descripción

class **DOMElement** {

DOMAttr **setAttributeNode** (DOMAttr attr)

}

Adds new attribute node *attr* to element.

Lista de parámetros

attr

The attribute node.

Valores retornados

Returns old node if the attribute has been replaced or **`NULL`**.

Exceptions

`DOM_NO_MODIFICATION_ALLOWED_ERR`

Raised if the node is readonly.

Ver también

DOMElement->hasAttribute()
DOMElement->getAttributeNode()
DOMElement->removeAttributeNode
()

DOMElement->setAttributeNodeNS()

DOMElement->setAttributeNodeNS() -- Adds new attribute node to element

Descripción

class **DOMElement** {

DOMAttr **setAttributeNodeNS** (DOMAttr attr)

}

Adds new attribute node *attr* to element.

Lista de parámetros

name

The attribute node.

Valores retornados

Returns the old node if the attribute has been replaced.

Exceptions

DOM_NO_MODIFICATION_ALLOWED_ERR

Raised if the node is readonly.

Ver también

DOMElement->hasAttributeNS()
DOMElement->getAttributeNodeNS()
DOMElement->removeAttributeNode
()

DOMElement->setAttributeNS()

DOMElement->setAttributeNS() -- Adds new attribute

Descripción

class **DOMElement** {

void **setAttributeNS** (string namespaceURI, string qualifiedName, string value)

}

Sets an attribute with namespace *namespaceURI* and name *name* to the given value. If the attribute does not exist, it will be created.

Lista de parámetros

namespaceURI

The namespace URI.

qualifiedName

The qualified name of the attribute, as *prefix:tagname*.

value

The value of the attribute.

Valores retornados

No value is returned.

Exceptions

`DOM_NO_MODIFICATION_ALLOWED_ERR`

Raised if the node is readonly.

`DOM_NAMESPACE_ERR`

Raised if *qualifiedName* is a malformed qualified name, or if *qualifiedName* has a prefix and *namespaceURI* is **`NULL`**.

Ver también

DOMElement->hasAttributeNS()
DOMElement->getAttributeNS()
DOMElement->removeAttributeNS()

DOMImplementation->createDocument()

DOMImplementation->createDocument() -- Creates a DOMDocument object of the specified type with its document element

Descripción

class **DOMImplementation** {

DOMDocument **createDocument** ([string namespaceURI [, string qualifiedName [, DOMDocumentType doctype]]])

}

Creates a **DOMDocument** object of the specified type with its document element.

Lista de parámetros

namespaceURI

The namespace URI of the document element to create.

qualifiedName

The qualified name of the document element to create.

doctype

The type of document to create or **NULL**.

Valores retornados

A new **DOMDocument** object. If *namespaceURI*, *qualifiedName*, and *doctype* are null, the returned **DOMDocument** is empty with no document element

Exceptions

DOM_WRONG_DOCUMENT_ERR

Raised if *doctype* has already been used with a different document or was created from a different implementation.

DOM_NAMESPACE_ERR

Raised if there is an error with the namespace, as determined by *namespaceURI* and *qualifiedName*.

Ver también

DOMDocument->__construct()
DOMImplementation->createDocumentType
()

DOMImplementation->createDocumentType()

DOMImplementation->createDocumentType() -- Creates an empty DOMDocumentType object

Descripción

class **DOMImplementation** {

DOMDocumentType **createDocumentType** ([string qualifiedName [, string publicId [, string systemId]]])

}

Creates an empty **DOMDocumentType** object. Entity declarations and notations are not made available. Entity reference expansions and default attribute additions do not occur.

Lista de parámetros

qualifiedName

The qualified name of the document type to create.

publicId

The external subset public identifier.

systemId

The external subset system identifier.

Valores retornados

A new **DOMDocumentType** node with its *ownerDocument* set to **NULL**.

Exceptions

DOM_NAMESPACE_ERR

Raised if there is an error with the namespace, as determined by *qualifiedName*.

Ver también

DOMImplementation->createDocument
()

DOMImplementation->hasFeature()

DOMImplementation->hasFeature() -- Test if the DOM implementation implements a specific feature

Descripción

class **DOMImplementation** {

bool **hasFeature** (string feature, string version)

}

Test if the DOM implementation implements a specific *feature*.

You can find a list of all features in the Conformance section of the DOM specification.

Lista de parámetros

feature

The feature to test.

version

The version number of the *feature* to test. In level 2, this can be either *2.0* or *1.0*.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplos

Ejemplo 1. Testing your DOM Implementation

```
<?php

$features = array(
  'Core'           => 'Core module',
  'XML'            => 'XML module',
  'HTML'           => 'HTML module',
  'Views'          => 'Views module',
  'Stylesheets'    => 'Style Sheets module',
  'CSS'            => 'CSS module',
  'CSS2'           => 'CSS2 module',
  'Events'         => 'Events module',
  'UIEvents'       => 'User interface Events module',
  'MouseEvents'    => 'Mouse Events module',
  'MutationEvents' => 'Mutation Events module',
  'HTMLEvents'     => 'HTML Events module',
  'Range'          => 'Range module',
  'Traversal'      => 'Traversal module'
);

foreach ($features as $key => $name) {
  if (DOMImplementation::hasFeature($key, '2.0')) {
    echo "Has feature $name\n";
  } else {
    echo "Missing feature $name\n";
  }
}

?>
```

Ver también

DOMNode->isSupported()

DOMNamedNodeMap->getNamedItem()

DOMNamedNodeMap->getNamedItem() -- Retrieves a node specified by name

Descripción

class **DOMNamedNodeMap** {

DOMNode **getNamedItem** (string name)

}

Retrieves a node specified by its *nodeName*.

Lista de parámetros

name

The *nodeName* of the node to retrieve.

Valores retornados

A node (of any type) with the specified *nodeName*, or **`NULL`** if no node is found.

Ver también

DOMNamedNodeMap->getNamedItemNS()

DOMNamedNodeMap->getNamedItemNS()

DOMNamedNodeMap->getNamedItemNS() -- Retrieves a node specified by local name and namespace URI

Descripción

class **DOMNamedNodeMap** {

DOMNode **getNamedItemNS** (string namespaceURI, string localName)

}

Retrieves a node specified by *localName* and *namespaceURI*.

Lista de parámetros

namespaceURI

The namespace URI of the node to retrieve.

localName

The local name of the node to retrieve.

Valores retornados

A node (of any type) with the specified local name and namespace URI, or **`NULL`** if no node is found.

Ver también

DOMNamedNodeMap->getNamedItem
()

DOMNamedNodeMap->item()

DOMNamedNodeMap->item() -- Retrieves a node specified by index

Descripción

class **DOMNamedNodeMap** {

DOMNode **item** (int index)

}

Retrieves a node specified by *index* within the **DOMNamedNodeMap** object.

Lista de parámetros

index

Index into this map.

Valores retornados

The node at the *index*th position in the map, or **`NULL`** if that is not a valid index (greater than or equal to the number of nodes in this map).

DOMNode->appendChild()

DOMNode->appendChild() -- Adds new child at the end of the children

Descripción

class **DOMNode** {

DOMNode **appendChild** (DOMNode newnode)

}

This functions appends a child to an existing list of children or creates a new list of children. The child can be created with e.g. DOMDocument->createElement(), DOMDocument->createTextNode() etc. or simply by using any other node.

Lista de parámetros

newnode

The appended child.

Valores retornados

The node added.

Exceptions

`DOM_NO_MODIFICATION_ALLOWED_ERR`

Raised if this node is readonly or if the previous parent of the node being inserted is readonly.

DOM_HIERARCHY_REQUEST_ERR

Raised if this node is of a type that does not allow children of the type of the *newnode* node, or if the node to append is one of this node's ancestors or this node itself.

DOM_WRONG_DOCUMENT_ERR

Raised if *newnode* was created from a different document than the one that created this node.

Ejemplos

The following example will add a new element node to a fresh document.

Ejemplo 1. Adding a child

```
<?php

$doc = new DOMDocument;

$node = $doc->createElement("para");
$newnode = $doc->appendChild($node);

echo $doc->saveXML();
?>
```

Ver también

DOMNode->removeChild
()
DOMNode->replaceChild()

DOMNode->cloneNode()

DOMNode->cloneNode() -- Clones a node

Descripción

class **DOMNode** {

DOMNode **cloneNode** ([bool deep])

}

Creates a copy of the node.

Lista de parámetros

deep

Indicates whether to copy all descendant nodes. This parameter is defaulted to **FALSE**.

Valores retornados

The cloned node.

DOMNode->hasAttributes()

DOMNode->hasAttributes() -- Checks if node has attributes

Descripción

class **DOMNode** {

bool **hasAttributes** (void)

}

This method checks if the node has attributes. The tested node have to be an **`XML_ELEMENT_NODE`**.

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Ver también

DOMNode->hasChildNodes()

DOMNode->hasChildNodes()

DOMNode->hasChildNodes() -- Checks if node has children

Descripción

class **DOMNode** {

bool **hasChildNodes** (void)

}

This function checks if the node has children.

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Ver también

DOMNode->hasAttributes
()

DOMNode->insertBefore()

DOMNode->insertBefore() -- Adds a new child before a reference node

Descripción

class **DOMNode** {

DOMNode **insertBefore** (DOMNode newnode [, DOMNode refnode])

}

This function inserts a new node right before the reference node. If you plan to do further modifications on the appended child you must use the returned node.

Lista de parámetros

newnode

The new node.

refnode

The reference node. If not supplied, *newnode* is appended to the children.

Valores retornados

The inserted node.

Exceptions

DOM_NO_MODIFICATION_ALLOWED_ERR

Raised if this node is readonly or if the previous parent of the node being inserted is readonly.

DOM_HIERARCHY_REQUEST_ERR

Raised if this node is of a type that does not allow children of the type of the *newnode* node, or if the node to append is one of this node's ancestors or this node itself.

DOM_WRONG_DOCUMENT_ERR

Raised if *newnode* was created from a different document than the one that created this node.

DOM_NOT_FOUND

Raised if *refnode* is not a child of this node.

DOMNode->isSameNode()

DOMNode->isSameNode() -- Indicates if two nodes are the same node

Descripción

class **DOMNode** {

bool **isSameNode** (DOMNode node)

}

This function indicates if two nodes are the same node. The comparison is *not* based on content

Lista de parámetros

node

The compared node.

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

DOMNode->isSupported()

DOMNode->isSupported() -- Checks if feature is supported for specified version

Descripción

class **DOMNode** {

bool **isSupported** (string feature, string version)

}

Checks if the asked *feature* is supported for the specified *version*.

Lista de parámetros

feature

The feature to test. See the example of DOMImplementation->hasFeature() for a list of features.

version

The version number of the *feature* to test.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ver también

DOMImplementation->hasFeature
()

DOMNode->lookupNamespaceURI()

DOMNode->lookupNamespaceURI() -- Gets the namespace URI of the node based on the prefix

Descripción

class **DOMNode** {

string **lookupNamespaceURI** (string prefix)

}

Gets the namespace URI of the node based on the *prefix*.

Lista de parámetros

prefix

The prefix of the namespace.

Valores retornados

The namespace URI of the node.

Ver también

DOMNode->lookupPrefix
()

DOMNode->lookupPrefix()

DOMNode->lookupPrefix() -- Gets the namespace prefix of the node based on the namespace URI

Descripción

class **DOMNode** {

string **lookupPrefix** (string namespaceURI)

}

Gets the namespace prefix of the node based on the namespace URI.

Lista de parámetros

namespaceURI

The namespace URI.

Valores retornados

The prefix of the namespace.

Ver también

DOMNode->lookupNamespaceURI()

DOMNode->normalize()

DOMNode->normalize() -- Normalizes the node

Descripción

class **DOMNode** {

void **normalize** (void)

}

Normalizes the node.

Valores retornados

No value is returned.

Ver también

DOMDocument->normalize
()

DOMNode->removeChild()

DOMNode->removeChild() -- Removes child from list of children

Descripción

class **DOMNode** {

DOMNode **removeChild** (DOMNode oldnode)

}

This functions removes a child from a list of children.

Lista de parámetros

oldnode

The removed child.

Valores retornados

If the child could be removed the functions returns the old child.

Exceptions

DOM_NO_MODIFICATION_ALLOWED_ERR

Raised if this node is readonly.

DOM_NOT_FOUND

Raised if *oldnode* is not a child of this node.

Ejemplos

The following example will delete the chapter element of our XML document.

Ejemplo 1. Removing a child

```
<?php

$doc = new DOMDocument;
$doc->load('book.xml');

$book = $doc->documentElement;

// we retrieve the chapter and remove it from the book
$chapter = $book->getElementsByTagName('chapter')->item(0);
$oldchapter = $book->removeChild($chapter);

echo $doc->saveXML();
?>
```

El resultado del ejemplo seria:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook XML V4.1.2//EN" "http://www.oasis-open.org
<book id="listing">
 <title>My lists</title>

</book>
```

Ver también

DOMNode->appendChild()
DOMNode->replaceChild()

DOMNode->replaceChild()

DOMNode->replaceChild() -- Replaces a child

Descripción

class **DOMNode** {

DOMNode **replaceChild** (DOMNode newnode, DOMNode oldnode)

}

This function replaces the child *oldnode* with the passed new node. If the new node is already a child it will not be added a second time. If the replacement succeeds the old node is returned.

Lista de parámetros

newnode

The new node.

oldnode

The old node.

Valores retornados

The old node or **FALSE** if an error occur.

Exceptions

DOM_NO_MODIFICATION_ALLOWED_ERR

Raised if this node is readonly or if the previous parent of the node being inserted is readonly.

DOM_HIERARCHY_REQUEST_ERR

Raised if this node is of a type that does not allow children of the type of the *newnode* node, or if the node to put in is one of this node's ancestors or this node itself.

DOM_WRONG_DOCUMENT_ERR

Raised if *newnode* was created from a different document than the one that created this node.

DOM_NOT_FOUND

Raised if *oldnode* is not a child of this node.

Ver también

DOMNode->appendChild()
DOMNode->removeChild
()

DOMNodelist->item()

DOMNodelist->item() -- Retrieves a node specified by index

Descripción

class **DOMNodeList** {

DOMNode **item** (int index)

}

Retrieves a node specified by *index* within the **DOMNodeList** object.

> **Sugerencia:** If you need to know the number of nodes in the collection, use the *length* property of the **DOMNodeList** object.

Lista de parámetros

index

> Index of the node into the collection.

Valores retornados

The node at the *index*th position in the **DOMNodeList**, or **`NULL`** if that is not a valid index.

Ejemplos

Ejemplo 1. Traversing all the entries of the table

```
<?php

$doc = new DOMDocument;
$doc->load('book.xml');

$items = $doc->getElementsByTagName('entry');

for ($i = 0; $i < $items->length; $i++) {
    echo $items->item($i)->nodeValue . "\n";
}

?>
```

Alternatively, you can use foreach, which is a much more convenient way:

```php
<?php

foreach ($items as $item) {
    echo $item->nodeValue . "\n";
}

?>
```

El resultado del ejemplo seria:

```
Title
Author
Language
ISBN
The Grapes of Wrath
John Steinbeck
en
0140186409
The Pearl
John Steinbeck
en
014017737X
Samarcande
Amine Maalouf
fr
2253051209
```

DOMText->isWhitespaceInElementContent()

DOMText->isWhitespaceInElementContent() -- Indicates whether this text node contains whitespace

Descripción

class **DOMText** {

bool **isWhitespaceInElementContent** (void)

}

Indicates whether this text node contains whitespace. The text node is determined to contain whitespace in element content during the load of the document.

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

DOMText->splitText()

DOMText->splitText() -- Breaks this node into two nodes at the specified offset

Descripción

class **DOMText** {

DOMText **splitText** (int offset)

}

Breaks this node into two nodes at the specified *offset*, keeping both in the tree as siblings.

After being split, this node will contain all the content up to the *offset*. If the original node had a parent node, the new node is inserted as the next sibling of the original node. When the *offset* is equal to the length of this node, the new node has no data.

Lista de parámetros

offset

The offset at which to split, starting from 0.

Valores retornados

The new node of the same type, which contains all the content at and after the *offset*.

DOMXPath->__construct()

DOMXPath->__construct() -- Creates a new DOMXPath object

Descripción

class **DOMXPath** {

__construct (DOMDocument doc)

}

Creates a new **DOMXPath** object.

Lista de parámetros

doc

The **DOMDocument** associated with the **DOMXPath**.

DOMXPath->evaluate()

DOMXPath->evaluate() -- Evaluates the given XPath expression and returns a typed result if possible.

Descripción

class **DOMXPath** {

mixed **evaluate** (string expression [, DOMNode contextnode])

}

Executes the given XPath *expression* and returns a typed result if possible.

Lista de parámetros

expression

The XPath expression to execute.

contextnode

The optional *contextnode* can be specified for doing relative XPath queries. By default, the queries are relative to the root element.

Valores retornados

Returns a typed result if possible or a **DOMNodeList** containing all nodes matching the given XPath *expression*.

Ejemplos

Ejemplo 1. Getting the count of all the english books

```
<?php

$doc = new DOMDocument;

$doc->load('book.xml');

$xpath = new DOMXPath($doc);

$tbody = $doc->getElementsByTagName('tbody')->item(0);

// our query is relative to the tbody node
$query = 'count(row/entry[. = "en"])';

$entries = $xpath->evaluate($query, $tbody);
echo "There are $entries english books\n";

?>
```

El resultado del ejemplo seria:

```
There are 2 english books
```

Ver también

DOMXPath->query()

DOMXPath->query()

DOMXPath->query() -- Evaluates the given XPath expression

Descripción

class **DOMXPath** {

DOMNodeList **query** (string expression [, DOMNode contextnode])

}

Executes the given XPath *expression*.

Lista de parámetros

expression

The XPath expression to execute.

contextnode

The optional *contextnode* can be specified for doing relative XPath queries. By default, the queries are relative to the root element.

Valores retornados

Returns a **DOMNodeList** containing all nodes matching the given XPath *expression*. Any expression which do not return nodes will return an empty **DOMNodeList**.

Ejemplos

Ejemplo 1. Getting all the english books

```
<?php

$doc = new DOMDocument;

// We don't want to bother with white spaces
$doc->preserveWhiteSpace = false;

$doc->Load('book.xml');

$xpath = new DOMXPath($doc);

// We starts from the root element
$query = '//book/chapter/para/informaltable/tgroup/tbody/row/entry[. = "en"]';

$entries = $xpath->query($query);

foreach ($entries as $entry) {
    echo "Found {$entry->previousSibling->previousSibling->nodeValue}," .
         " by {$entry->previousSibling->nodeValue}\n";
}
?>
```

El resultado del ejemplo seria:

```
Found The Grapes of Wrath, by John Steinbeck
Found The Pearl, by John Steinbeck
```

We can also use the *contextnode* parameter to shorten our expression:

```php
<?php

$doc = new DOMDocument;
$doc->preserveWhiteSpace = false;

$doc->load('book.xml');

$xpath = new DOMXPath($doc);

$tbody = $doc->getElementsByTagName('tbody')->item(0);

// our query is relative to the tbody node
$query = 'row/entry[. = "en"]';

$entries = $xpath->query($query, $tbody);

foreach ($entries as $entry) {
    echo "Found {$entry->previousSibling->previousSibling->nodeValue}," .
         " by {$entry->previousSibling->nodeValue}\n";
}
?>
```

Ver también

DOMXPath->evaluate
()

DOMXPath->registerNamespace()

DOMXPath->registerNamespace() -- Registers the namespace with the DOMXpath object

Descripción

class **DOMXPath** {

bool **registerNamespace** (string prefix, string namespaceURI)

}

Registers the *namespaceURI* and *prefix* with the DOMXpath object.

Lista de parámetros

prefix

The prefix.

namespaceURI

The URI of the namespace.

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

dom_import_simplexml

(PHP 5)

dom_import_simplexml -- Gets a DOMElement object from a SimpleXMLElement object

Descripción

DOMElement **dom_import_simplexml** (SimpleXMLElement node)

This function takes the node *node* of class SimpleXML and makes it into a **DOMElement** node. This new object can then be used as a native **DOMElement** node.

Lista de parámetros

node

The **SimpleXMLElement** node.

Valores retornados

The **DOMElement** node added or **FALSE** if any errors occur.

Ejemplos

Ejemplo 1. Import SimpleXML into DOM with dom_import_simplexml()

```
<?php

$sxe = simplexml_load_string('<books><book><title>blah</title></book></books>');

if ($sxe === false) {
    echo 'Error while parsing the document';
    exit;
}

$dom_sxe = dom_import_simplexml($sxe);
if (!$dom_sxe) {
    echo 'Error while converting XML';
    exit;
}

$dom = new DOMDocument('1.0');
$dom_sxe = $dom->importNode($dom_sxe, true);
$dom_sxe = $dom->appendChild($dom_sxe);

echo $dom->saveXML();

?>
```

Ver también

simplexml_import_dom
()

XXVIII. Funciones DOM XML

Introducción

Aviso
Esta extensión es *EXPERIMENTAL*. Esto significa que el comportamiento de esta extensión, los nombre de sus funciones y en definitiva TODO lo documentado sobre esta extensión, puede cambiar en una futura versión de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

La extensión DOM XML ha sido re-estructurada en PHP 4.3.0 para mayor compatibilidad con el estándar DOM. La extensión aun contiene varias funciones viejas, pero ellas ya no deben ser usadas. En particular, las funciones que no son orientadas a objetos deben evitarse.

La extensión le permite operar sobre un documento XML con la API DOM. También ofrece una función **domxml_xmltree()** para convertir el documento XML completo en un árbol de objetos PHP. Actualmente, este árbol debe ser considerado como de sólo-escritura - es posible modificarlo, pero tal cosa no tendría sentido ya que **DomDocument_dump_mem()** no puede aplicarse sobre él. Por lo tanto, si desea leer un archivo XML y escribir una versión modificada, use **DomDocument_create_element()**, **DomDocument_create_text_node()**, **set_attribute()**, etc. y finalmente la función **DomDocument_dump_mem()**.

> **Nota:** Esta extension ha sido quitada a partir de PHP 5 y se puede encontrar en el repositorio PECL.
>
> **Nota:** Si necesita soporte DOM XML con PHP 5, puede usar la extensión DOM.

Requirimientos

Esta extensión hace uso de la biblioteca GNOME XML. Descargue e instale esta biblioteca. Necesita por lo menos libxml-2.4.14. Para usar las características DOM XSLT, puede usar la biblioteca libxslt y las adiciones EXSLT de http://www.exslt.org/. Descargue e instale estas bibliotecas si planea usar las características XSLT (y las mejoras). Necesita por lo menos libxslt-1.0.18.

Instalación

Esta extension PECL no esta ligada a PHP. Mas informacion sobre nuevos lanzamientos, descargas ficheros de fuentes, informacion sobre los responsables asi como un 'CHANGELOG', se puede encontrar aqui: http://pecl.php.net/package/domxml.

En PHP 4 la fuente de las extensiones PECL pueden encontrarse en el directorio `ext/` que se existe en las fuentes de PHP o en el enlace PECL de arriba. Esta extensión se encuentra disponible únicamente si PHP fue configurado con *--with-dom[=DIR]*. Agregue *--with-dom-xslt[=DIR]* para incluir soporte para DOM XSLT. DIR es el directorio de instalación de libxslt. Agregue *--with-dom-exslt[=DIR]* para incluir soporte para DOM EXSLT, en donde DIR es el directorio de instalación de libexslt.

Los usuarios de windows deben habilitar `php_domxml.dll` al interior de `php.ini` para usar

estas funciones. En PHP 4, esta DLL se encuentra en el directorio `extensions/` que existe en los binarios de PHP para Windows. Podeis descargar esta DLL de las extensiones PECL desde la pagina PHP Downloads o desde http://snaps.php.net/. Asimismo, hay una DLL adicional que debe estar disponible para su PATH de sistema para que ésta extensión trabaje. En PHP 4 esta ruta está en el directorio `dlls/`. Su nombre: Para PHP <= 4.2.0, es `libxml2.dll`. Para PHP >= 4.3.0, es `iconv.dll`. Y a partir de PHP 5.0.0, iconv se encuentra compilado con sus binarios Windows de PHP por defecto, así que no se necesitan archivos DLL adicionales.

Funciones obsoletas

Existen varias funciones que no tienen lugar en el estándar DOM y no deberían seguir siendo usadas. Estas funciones son listadas en la siguiente tabla. La función **DomNode_append_child()** ha cambiado su comportamiento. Ahora agrega un hijo y no un hermano. Si esto afecta su aplicación, use la función **DomNode_append_sibling()**, la cual no hace parte del conjunto DOM.

Tabla 1. Funciones obsoletas y sus reemplazos

Función antigua	Función nueva
xmldoc	**domxml_open_mem()**
xmldocfile	**domxml_open_file()**
domxml_new_xmldoc	**domxml_new_doc()**
domxml_dump_mem	**DomDocument_dump_mem()**
domxml_dump_mem_file	**DomDocument_dump_file()**
DomDocument_dump_mem_file	**DomDocument_dump_file()**
DomDocument_add_root	**DomDocument_create_element()** seguido por **DomNode_append_child()**
DomDocument_dtd	**DomDocument_doctype()**
DomDocument_root	**DomDocument_document_element()**
DomDocument_children	**DomNode_child_nodes()**
DomDocument_imported_node	No hay reemplazo.
DomNode_add_child	Crear un nuevo nodo, p.ej. con **DomDocument_create_element()** y agregarlo con **DomNode_append_child()**.
DomNode_children	**DomNode_child_nodes()**
DomNode_parent	**DomNode_parent_node()**
DomNode_new_child	Crear un nuevo nodo, p.ej. con **DomDocument_create_element()** y agregarlo con **DomNode_append_child()**.
DomNode_set_content	Crear un nuevo nodo, p.ej. con **DomDocument_create_text_node()** y agregarlo con **DomNode_append_child()**.
DomNode_get_content	El contenido es solo un nodo de texto y puede consultarse con **DomNode_child_nodes()**.

Función antigua	Función nueva
DomNode_set_content	El contenido es solo un nodo de texto y puede ser agregado con **DomNode_append_child()**.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

Tabla 2. Constantes XML

Constante	Valor	Descripción
XML_ELEMENT_NODE (integer)	1	El nodo es un elemento
XML_ATTRIBUTE_NODE (integer)	2	El nodo es un atributo
XML_TEXT_NODE (integer)	3	El nodo es un segmento de texto
XML_CDATA_SECTION_NODE (integer)	4	
XML_ENTITY_REF_NODE (integer)	5	
XML_ENTITY_NODE (integer)	6	El nodo es una entidad como
XML_PI_NODE (integer)	7	El nodo es una instrucción de procesamiento
XML_COMMENT_NODE (integer)	8	El nodo es un comentario
XML_DOCUMENT_NODE (integer)	9	El nodo es un documento
XML_DOCUMENT_TYPE_NODE (integer)	10	
XML_DOCUMENT_FRAG_NODE (integer)	11	
XML_NOTATION_NODE (integer)	12	
XML_GLOBAL_NAMESPACE (integer)	1	
XML_LOCAL_NAMESPACE (integer)	2	
XML_HTML_DOCUMENT_NODE (integer)		
XML_DTD_NODE (integer)		
XML_ELEMENT_DECL_NODE (integer)		
XML_ATTRIBUTE_DECL_NODE (integer)		
XML_ENTITY_DECL_NODE (integer)		
XML_NAMESPACE_DECL_NODE (integer)		
XML_ATTRIBUTE_CDATA (integer)		
XML_ATTRIBUTE_ID (integer)		
XML_ATTRIBUTE_IDREF (integer)		
XML_ATTRIBUTE_IDREFS (integer)		
XML_ATTRIBUTE_ENTITY (integer)		

Constante	Valor	Descripción
XML_ATTRIBUTE_NMTOKEN (integer)		
XML_ATTRIBUTE_NMTOKENS (integer)		
XML_ATTRIBUTE_ENUMERATION (integer)		
XML_ATTRIBUTE_NOTATION (integer)		
XPATH_UNDEFINED (integer)		
XPATH_NODESET (integer)		
XPATH_BOOLEAN (integer)		
XPATH_NUMBER (integer)		
XPATH_STRING (integer)		
XPATH_POINT (integer)		
XPATH_RANGE (integer)		
XPATH_LOCATIONSET (integer)		
XPATH_USERS (integer)		
XPATH_NUMBER (integer)		

Clases

La API del módulo sigue el estándar DOM de Nivel 2 tan fielmente como es posible. Por consiguiente, la API es completamente orientada a objetos. Es una buena idea tener el estándar DOM a la mano cuando se usa este módulo. Aunque la API es orientada a objetos, existen varias funciones que pueden ser llamadas en una forma no orientada a objetos, pasando el objeto sobre el que debe operarse como primer argumento. Estas funciones existen principalmente para conservar la compatibilidad con versiones anteriores de esta extensión, y no deberían ser usadas cuando se creen nuevos scripts.

Esta API difiere de la API DOM oficial en dos formas. Primero, todos los atributos de clase son implementados como funciones con el mismo nombre. En segundo lugar, los nombres de funciones siguen la convención de nombres de PHP. esto quiere decir que una función DOM llamada lastChild() será escrita como last_child().

Este módulo define un número de clases, que son listados - incluyendo sus métodos - en las siguientes tablas. Las clases con un ñalente en el estándar DOM son llamadas DOMxxx.

Tabla 3. Lista de clases

Nombre de clase	Clases padre
DomAttribute	DomNode
DomCData	DomNode
DomComment	DomCData : DomNode
DomDocument	DomNode
DomDocumentType	DomNode

Nombre de clase	Clases padre
DomElement	DomNode
DomEntity	DomNode
DomEntityReference	DomNode
DomProcessingInstruction	DomNode
DomText	DomCData : DomNode
Parser	Por el momento aun se llama DomParser
XPathContext	

Tabla 4. Clase DomDocument (DomDocument : DomNode)

Nombre de método	Nombre de función	Anotación
doctype	**DomDocument_doctype()**	
document_element	**DomDocument_document_element()**	
create_element	**DomDocument_create_element()**	
create_text_node	**DomDocument_create_text_node()**	
create_comment	**DomDocument_create_comment()**	
create_cdata_section	**DomDocument_create_cdata_section()**	
create_processing_instruction	**DomDocument_create_processing_instruction()**	
create_attribute	**DomDocument_create_attribute()**	
create_entity_reference	**DomDocument_create_entity_reference()**	
get_elements_by_tagname	**DomDocument_get_elements_by_tagname()**	
get_element_by_id	**DomDocument_get_element_by_id()**	
dump_mem	**DomDocument_dump_mem()**	no hace parte del estándar DOM
dump_file	**DomDocument_dump_file()**	no hace parte del estándar DOM
html_dump_mem	**DomDocument_html_dump_mem()**	no hace parte del estándar DOM
xpath_init	xpath_init	no hace parte del estándar DOM
xpath_new_context	xpath_new_context	no hace parte del estándar DOM
xptr_new_context	xptr_new_context	no hace parte del estándar DOM

Tabla 5. Clase DomElement (DomElement : DomNode)

Nombre de método	Nombre de función	Anotación
tagname	DomElement_tagname()	
get_attribute	DomElement_get_attribute()	
set_attribute	DomElement_set_attribute()	
remove_attribute	DomElement_remove_attribute()	
get_attribute_node	DomElement_get_attribute_node()	
get_elements_by_tagname	DomElement_get_elements_by_tagname()	
has_attribute	DomElement_has_attribute()	

Tabla 6. DomNode class

Nombre de método	Anotación
DomNode_node_name()	
DomNode_node_value()	
DomNode_node_type()	
DomNode_last_child()	
DomNode_first_child()	
DomNode_child_nodes()	
DomNode_previous_sibling()	
DomNode_next_sibling()	
DomNode_parent_node()	
DomNode_owner_document()	
DomNode_insert_before()	
DomNode_append_child()	
DomNode_append_sibling()	No se encuentra en el estándar DOM. Esta función emula el comportamiento antiguo de DomNode_append_child().
DomNode_remove_child()	
DomNode_has_child_nodes()	

Nombre de método	Anotación
DomNode_has_attributes()	
DomNode_clone_node()	
DomNode_attributes()	
DomNode_unlink_node()	No se encuentra en el estándar DOM
DomNode_replace_node()	No se encuentra en el estándar DOM
DomNode_set_content()	No se encuentra en el estándar DOM, obsoleta
DomNode_get_content()	No se encuentra en el estándar DOM, obsoleta
DomNode_dump_node()	No se encuentra en el estándar DOM
DomNode_is_blank_node()	No se encuentra en el estándar DOM

Tabla 7. Clase DomAttribute (DomAttribute : DomNode)

Nombre de método		Anotación
name	DomAttribute_name()	
value	DomAttribute_value()	
specified	DomAttribute_specified()	

Tabla 8. Clase DomProcessingInstruction (DomProcessingInstruction : DomNode)

Nombre de método	Nombre de función	Anotación
target	DomProcessingInstruction_target()	
data	DomProcessingInstruction_data()	

Tabla 9. Clase Parser

Nombre de método	Nombre de función	Anotación
add_chunk	**Parser_add_chunk()**	
end	**Parser_end()**	

Tabla 10. Clase XPathContext

Nombre de método	Nombre de función	Anotación
eval	**XPathContext_eval()**	
eval_expression	**XPathContext_eval_expression ()**	
register_ns	**XPathContext_register_ns()**	

Tabla 11. Clase DomDocumentType (DomDocumentType : DomNode)

Nombre de método	Nombre de función	Anotación
name	**DomDocumentType_name()**	
entities	**DomDocumentType_entities()**	
notations	**DomDocumentType_notations()**	
public_id	**DomDocumentType_public_id()**	
system_id	**DomDocumentType_system_id()**	
internal_subset	**DomDocumentType_internal_subset ()**	

La clase DomDtd es derivada de DomNode. DomComment es derivada de DomCData.

Ejemplos

Varios ejemplos en esta referencia requieren una cadena XML. En lugar de repetir esta cadena en cada ejemplo, será puesta en un archivo el cual será incluido en cada ejemplo. Este archivo de inclusión es mostrado en la siguiente sección de ejemplo. Alternativamente, es posible crear un documento XML y leerlo con **DomDocument_open_file()**.

Ejemplo 1. Archivo de inclusión ejemplo.inc con una cadena XML

```
<?php
$cadena_xml = "<?xml version='1.0' standalone='yes'?>
<!DOCTYPE chapter SYSTEM '/share/sgml/Norman_Walsh/db3xml10/db3xml10.dtd'
[ <!ENTITY sp \"spanish\">
]>
<!-- lsfj  -->
<chapter language='en'><title language='en'>Title</title>
 <para language='ge'>
  &amp;sp;
  <!-- comment -->
  <informaltable ID='findme' language='&amp;sp;'>
   <tgroup cols='3'>
    <tbody>
     <row><entry>a1</entry><entry
morerows='1'>b1</entry><entry>c1</entry></row>
<row><entry>a2</entry><entry>c2</entry></row>
     <row><entry>a3</entry><entry>b3</entry><entry>c3</entry></row>
    </tbody>
   </tgroup>
  </informaltable>
 </para>
</chapter>";
?>
```

Tabla de contenidos

DomAttribute->specified -- Revisa si el atributo está especificado
DomAttribute->value -- Devuelve el valor del atributo
DomDocument->add_root -- Agrega un nodo raíz [obsoleto]
DomDocument->create_attribute -- Crear un nuevo atributo
DomDocument->create_cdata_section -- Crear un nuevo nodo cdata
DomDocument->create_comment -- Crea un nuevo nodo de comentario
DomDocument->create_element_ns -- Crea un nuevo nodo tipo elemento con un espacio de nombres asociado
DomDocument->create_element -- Crear un nuevo nodo de tipo elemento
DomDocument->create_entity_reference --
DomDocument->create_processing_instruction -- Crea un nuevo nodo PI
DomDocument->create_text_node -- Crear un nuevo nodo de texto
DomDocument->doctype -- Devuelve el tipo de documento
DomDocument->document_element -- Devuelve el nodo del elemento raíz
DomDocument->dump_file -- Vuelca el árbol XML interno de vuelta a un archivo
DomDocument->dump_mem -- Vuelca el árbol XML interno de vuelta a una cadena
DomDocument->get_element_by_id -- Busca un elemento con cierto id
DomDocument->get_elements_by_tagname --
DomDocument->html_dump_mem -- Vuelca el árbol XML interno de vuelta a una cadena como HTML
DomDocument->xinclude -- Reemplaza sentencias XInclude en un Objeto DomDocument
DomDocumentType->entities -- Devuelve una lista de entidades
DomDocumentType->internal_subset -- Devuelve el sub-conjunto interno
DomDocumentType->name -- Devuelve el nombre del tipo de documento
DomDocumentType->notations -- Devuelve una lista de notaciones
DomDocumentType->public_id -- Devuelve el id público del tipo de documento
DomDocumentType->system_id -- Devuelve el id de sistema del tipo de documento
DomElement->get_attribute_node -- Devuelve el nodo del atributo dado
DomElement->get_attribute -- Devuelve el valor del atributo dado
DomElement->get_elements_by_tagname -- Obtiene elementos por el nombre de etiqueta
DomElement->has_attribute -- Verifica si un atributo existe en el nodo actual
DomElement->remove_attribute -- Elimina un atributo
DomElement->set_attribute -- Define el valor de un atributo
DomElement->tagname -- Devuelve el nombre del elemento actual
DomNode->add_namespace -- Agrega una declaración de espacio de nombres a un nodo
DomNode->append_child -- Agrega un nuevo hijo al final del grupo de hijos
DomNode->append_sibling -- Agrega un nuevo hermano a un nodo
DomNode->attributes -- Devuelve la lista de atributos
DomNode->child_nodes -- Devuelve los hijos del nodo
DomNode->clone_node -- Clona un nodo
DomNode->dump_node -- Vuelca un nodo único
DomNode->first_child -- Devuelve el primer hijo del nodo
DomNode->get_content -- Obtiene el contenido del nodo
DomNode->has_attributes -- Verifica si un nodo tiene atributos
DomNode->has_child_nodes -- Verifica si el nodo tiene hijos
DomNode->insert_before -- Inserta un nodo nuevo como hijo
DomNode->is_blank_node -- Verifica si el nodo está en blanco
DomNode->last_child -- Devuelve el último hijo del nodo
DomNode->next_sibling -- Devuelve el siguiente hermano del nodo
DomNode->node_name -- Devuelve el nombre del nodo
DomNode->node_type -- Devuelve el tipo de nodo
DomNode->node_value -- Devuelve el valor de un nodo
DomNode->owner_document -- Devuelve el documento al que este nodo pertenece
DomNode->parent_node -- Devuelve el padre del nodo
DomNode->prefix -- Devuelve el prefijo de espacio de nombres del nodo

DomNode->previous_sibling -- Devuelve el hermano anterior del nodo
DomNode->remove_child -- Elimina un hijo de una lista de hijos
DomNode->replace_child -- Reemplaza un hijo
DomNode->replace_node -- Reemplaza el nodo
DomNode->set_content -- Define el contenido del nodo
DomNode->set_name -- Define el nombre del nodo
DomNode->set_namespace -- Define el espacio de nombres de un nodo
DomNode->unlink_node -- Elimina el nodo
DomProcessingInstruction->data -- Devuelve los datos de un nodo pi
DomProcessingInstruction->target -- Devuelve el destino de un nodo pi
DomXsltStylesheet->process -- Aplica la Transformación XSLT sobre un objeto DomDocument
DomXsltStylesheet->result_dump_file -- Vuelca el resultado de una Transformación XSLT a un archivo
DomXsltStylesheet->result_dump_mem -- Vuelca el resultado de una Transformación XSLT de vuelta a una cadena
domxml_new_doc -- Crea un nuevo documento XML vacío
domxml_open_file -- Crea un objeto DOM a partir de un archivo XML
domxml_open_mem -- Crea un objeto DOM desde un documento XML
domxml_version -- Obtiene la versión de la biblioteca XML
domxml_xmltree -- Crea un árbol de objetos PHP a partir de un documento XML
domxml_xslt_stylesheet_doc -- Crea un Objeto DomXsltStylesheet a partir de un Objeto DomDocument
domxml_xslt_stylesheet_file -- Crea un Objeto DomXsltStylesheet a partir de un documento XSL en un archivo
domxml_xslt_stylesheet -- Crea un Objeto DomXsltStylesheet desde un documento XML en una cadena
xpath_eval_expression -- Evalúa la Ruta de Ubicación XPath en la cadena entregada
xpath_eval -- Evalúa la Ruta de Ubicación XPatch en la cadena dada
xpath_new_context -- Crea un nuevo contexto xpath
xptr_eval -- Evalúa la Ruta de Ubicación XPtr en la cadena dada
xptr_new_context -- Crea un nuevo Contexto XPath

DomAttribute->name

(no version information, might be only in CVS)

DomAttribute->name -- Devuelve el nombre de un atributo

Descripción

string **DomAttribute->name** (void)

Esta función devuelve el nombre del atributo.

Vea también **domattribute_value()** para encontrar un ejemplo.

DomAttribute->specified

(no version information, might be only in CVS)

DomAttribute->specified -- Revisa si el atributo está especificado

Descripción

bool **DomAttribute->specified** (void)

Revise el estándar DOM para una explicación detallada.

DomAttribute->value

(no version information, might be only in CVS)

DomAttribute->value -- Devuelve el valor del atributo

Descripción

mixed **DomAttribute->value** (void)

Esta función devuelve el valor del atributo.

Ejemplo 1. Obtener todos los atributos de un nodo

```
<?php

include("ejemplo.inc");

if (!$dom = domxml_open_mem($cadena_xml)) {
     echo "Ocurri&oacute; un error al interpretar el documento\n";
     exit;
}

$raiz = $dom->document_element();
$atrs = $raiz->attributes();

echo 'Atributos de ' . $raiz->node_name() . "\n";
foreach ($atrs as $atributo) {
     echo ' - ' . $atributo->name . ' : ' . $attribute->value . "\n";
}

?>
```

El resultado del ejemplo seria:

```
Atributos de chapter
 - language : en
```

Vea también **domattribute_name()**.

DomDocument->add_root

(no version information, might be only in CVS)

DomDocument->add_root -- Agrega un nodo raíz [obsoleto]

Descripción

object **DomDocument->add_root** (string nombre)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Agrega un nodo de elemento raíz a un documento dom y devuelve el nuevo nodo. El nombre del elemento es dado en el parámetro pasado.

Ejemplo 1. Creación de una cabecera para un documento HTML simple

```
<?php
$doc = domxml_new_doc("1.0");
$raiz = $doc->add_root("html");
$cabecera = $raiz->new_child("head", "");
$cabecera->new_child("title", "Hier der Titel");
echo htmlentities($doc->dump_mem());
?>
```

DomDocument->create_attribute

(no version information, might be only in CVS)

DomDocument->create_attribute -- Crear un nuevo atributo

Descripción

object **DomDocument->create_attribute** (string nombre, string valor)

Esta función devuelve una nueva instancia de clase **DomAttribute**. El nombre del atributo es el valor del primer parámetro. El valor del atributo es el valor del segundo parámetro. Este nodo no sera mostrado en el documento a no ser que sea introducido por ejemplo con **domnode_append_child()**.

El valor de retorno es **FALSE** si ocurre un error.

Vea también **domnode_append_child()**, **domdocument_create_element()**, **domdocument_create_text()**, **domdocument_create_cdata_section()**, **domdocument_create_processing_instruction()**, **domdocument_create_entity_reference()**, y **domnode_insert_before()**.

DomDocument->create_cdata_section

(no version information, might be only in CVS)

DomDocument->create_cdata_section -- Crear un nuevo nodo cdata

Descripción

object **DomDocument->create_cdata_section** (string contenido)

Esta función devuelve una nueva instancia de la clase **DomCData**. El contenido cdata es el valor del parámetro pasado. Este nodo no sera mostrado en el documento a no ser que sea introducido por ejemplo con **domnode_append_child()**.

El valor de retorno es **FALSE** si ocurre un error.

Vea también **domnode_append_child()**, **domdocument_create_element()**, **domdocument_create_text()**, **domdocument_create_attribute()**, **domdocument_create_processing_instruction()**, **domdocument_create_entity_reference()**, y **domnode_insert_before()**.

DomDocument->create_comment

(no version information, might be only in CVS)

DomDocument->create_comment -- Crea un nuevo nodo de comentario

Descripción

object **DomDocument->create_comment** (string contenido)

Esta función devuelve una nueva instancia de la clase **DomComment**. El contenido del comentario es el valor del parámetro pasado. Este nodo no sera mostrado en el documento a no ser que sea introducido por ejemplo con **domnode_append_child()**.

El valor de retorno es **FALSE** si ocurre un error.

Vea también **domnode_append_child()**, **domdocument_create_element()**, **domdocument_create_text()**, **domdocument_create_attribute()**, **domdocument_create_processing_instruction()**, **domdocument_create_entity_reference()**, y **domnode_insert_before()**.

DomDocument->create_element_ns

(no version information, might be only in CVS)

DomDocument->create_element_ns -- Crea un nuevo nodo tipo elemento con un espacio de nombres asociado

Descripción

object **DomDocument->create_element_ns** (string uri, string nombre [, string prefijo])

Esta función devuelve una nueva instancia de la clase **DomElement**. El nombre de etiqueta del elemento es el valor del parámetro *nombre* pasado. El URI del espacio de nombres es el valor del parámetro *uri* pasado. Si ya existe una declaración de espacio de nombres con la misma uri en el nodo raíz del documento, el prefijo de éste es tomado, de lo contrario tomará el que se provee en el parámetro opcional *prefijo* o genera uno aleatorio. Este nodo no sera mostrado en el documento a no ser que sea introducido por ejemplo con **domnode_append_child()**.

El valor de retorno es **FALSE** si ocurre un error.

Vea también **domdocument_create_element_ns()**, **domnode_add_namespace()**, **domnode_set_namespace()**, **domnode_append_child()**, **domdocument_create_text()**, **domdocument_create_comment()**, **domdocument_create_attribute()**,

domdocument_create_processing_instruction(), **domdocument_create_entity_reference()**, y **domnode_insert_before()**.

DomDocument->create_element

(no version information, might be only in CVS)

DomDocument->create_element -- Crear un nuevo nodo de tipo elemento

Descripción

object **DomDocument->create_element** (string nombre)

Esta función devuelve una nueva instancia de la clase **DomElement**. El nombre de la etiqueta del elemento es el valor del parámetro pasado. Este nodo no sera mostrado en el documento a no ser que sea introducido por ejemplo con **domnode_append_child()**.

El valor de retorno es **FALSE** si ocurre un error.

Vea también **domdocument_create_element_ns()**, **domnode_append_child()**, **domdocument_create_text()**, **domdocument_create_comment()**, **domdocument_create_attribute()**, **domdocument_create_processing_instruction()**, **domdocument_create_entity_reference()**, y **domnode_insert_before()**.

DomDocument->create_entity_reference

(no version information, might be only in CVS)

DomDocument->create_entity_reference --

Descripción

object **DomDocument->create_entity_reference** (string contenido)

Esta función devuelve una nueva instancia de la clase **DomEntityReference**. El contenido de la referencia a entidad es el valor del parámetro pasado. Este nodo no sera mostrado en el documento a no ser que sea introducido por ejemplo con **domnode_append_child()**.

El valor de retorno es **FALSE** si ocurre un error.

Vea también **domnode_append_child()**, **domdocument_create_element()**, **domdocument_create_text()**, **domdocument_create_cdata_section()**, **domdocument_create_processing_instruction()**, **domdocument_create_attribute()**, y **domnode_insert_before()**.

DomDocument->create_processing_instruction

(no version information, might be only in CVS)

DomDocument->create_processing_instruction -- Crea un nuevo nodo PI

Descripción

object **DomDocument->create_processing_instruction** (string contenido)

Esta función devuelve una nueva instancia de la clase **DomCData**. El contenido de la instrucción de procesamiento es el valor del parámetro pasado. Este nodo no sera mostrado en el documento a no ser que sea introducido por ejemplo con **domnode_append_child()**.

El valor de retorno es **FALSE** si ocurre un error.

Vea también **domnode_append_child()**, **domdocument_create_element()**, **domdocument_create_text()**, **domdocument_create_cdata_section()**, **domdocument_create_attribute()**, **domdocument_create_entity_reference()**, y **domnode_insert_before()**.

DomDocument->create_text_node

(no version information, might be only in CVS)

DomDocument->create_text_node -- Crear un nuevo nodo de texto

Descripción

object **DomDocument->create_text_node** (string contenido)

Esta función devuelve una nueva instancia de la clase **DomText**. El contenido del texto es el valor del parámetro pasado. Este nodo no sera mostrado en el documento a no ser que sea introducido por ejemplo con **domnode_append_child()**.

El valor de retorno es **FALSE** si ocurre un error.

Vea también **domnode_append_child()**, **domdocument_create_element()**, **domdocument_create_comment()**, **domdocument_create_text()**, **domdocument_create_attribute()**, **domdocument_create_processing_instruction()**, **domdocument_create_entity_reference()**, y **domnode_insert_before()**.

DomDocument->doctype

(no version information, might be only in CVS)

DomDocument->doctype -- Devuelve el tipo de documento

Descripción

object **DomDocument->doctype** (void)

Esta función devuelve un objeto de la clase **DomDocumentType**. En versiones de PHP anteriores a 4.3, esta era la clase **Dtd**, pero el estándar DOM no conoce tal clase.

Vea también los métodos de la clase **DomDocumentType**.

DomDocument->document_element

(no version information, might be only in CVS)

DomDocument->document_element -- Devuelve el nodo del elemento raíz

Descripción

object **DomDocument->document_element** (void)

Esta función devuelve el nodo de elemento raíz de un documento.

El siguiente ejemplo devuelve tan solo el elemento con nombre CHAPTER y lo imprime. El otro nodo -- el comentario -- no es devuelto.

Ejemplo 1. Recuperar el elemento raíz

```
<?php
include("ejemplo.inc");

if (!$dom = domxml_open_mem($cadena_xml)) {
  echo "Ha ocurrido un error al procesar el documento\n";
  exit;
}

$raiz = $dom->document_element();
print_r($raiz);
?>
```

Esto producirá la salida:

```
domelement Object
(
    [type] => 1
    [tagname] => chapter
    [0] => 6
    [1] => 137960648
)
```

DomDocument->dump_file

(no version information, might be only in CVS)

DomDocument->dump_file -- Vuelca el árbol XML interno de vuelta a un archivo

Descripción

string **DomDocument->dump_file** (string nombre_archivo [, bool modo_compresion [, bool formato]])

Crea un documento XML desde la representación dom. Esta función es llamada por lo general después de construir un nuevo documento dom desde ceros como en el ejemplo a continuación. El *formato* especifica si la salida debe tener un formato elegante, o no. El primer parámetro especifica el nombre del archivo y el segundo parámetro, si debe comprimirse o no.

Ejemplo 1. Creación de una cabecera de un documento HTML simple

```
<?php
$doc = domxml_new_doc("1.0");
$raiz = $doc->create_element("HTML");
$raiz = $doc->append_child($raiz);
$cabecera = $doc->create_element("HEAD");
$cabecera = $raiz->append_child($cabecera);
$titulo = $doc->create_element("TITLE");
$titulo = $cabecera->append_child($titulo);
$texto = $doc->create_text_node("Este es el t&iacute;tulo");
$texto = $titulo->append_child($texto);
$doc->dump_file("/tmp/prueba.xml", false, true);
?>
```

Vea también **domdocument_dump_mem()**, y **domdocument_html_dump_mem()**.

DomDocument->dump_mem

(no version information, might be only in CVS)

DomDocument->dump_mem -- Vuelca el árbol XML interno de vuelta a una cadena

Descripción

string **DomDocument->dump_mem** ([bool formato [, string codificacion]])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Crea un documento XML desde la representación dom. Esta función es llamada por lo general después de construir un nuevo documento dom desde ceros como en el ejemplo a continuación. El *formato* especifica si la salida debe tener un formato elegante, o no.

Ejemplo 1. Creación de una cabecera de un documento HTML simple

```
<?php
$doc = domxml_new_doc("1.0");
$raiz = $doc->create_element("HTML");
$raiz = $doc->append_child($raiz);
$cabecera = $doc->create_element("HEAD");
$cabecera = $raiz->append_child($cabecera);
$titulo = $doc->create_element("TITLE");
$titulo = $cabecera->append_child($titulo);
$texto = $doc->create_text_node("This is the title");
$texto = $titulo->append_child($texto);
echo "<PRE>";
echo htmlentities($doc->dump_mem(true));
echo "</PRE>";
?>
```

Nota: El primer parámetro fue agregado en PHP 4.3.0.

Vea también **domdocument_dump_file()**, y **domdocument_html_dump_mem()**.

DomDocument->get_element_by_id

(no version information, might be only in CVS)

DomDocument->get_element_by_id -- Busca un elemento con cierto id

Descripción

object **DomDocument->get_element_by_id** (string id)

Esta función es similar a **domdocument_get_elements_by_tagname()**, pero busca un elemento con un id dado. De acuerdo al estándar DOM, esto requiere un DTD que defina el atributo ID como de tipo ID, aunque la implementación actual simplemente realiza una búsqueda xpath por "//*[@ID = '%s']". Esto no es fiel al estándar DOM, el cual requiere que se devuelva un valor nulo si no se conoce cuál atributo es de tipo id. Es probable que este comportamiento sea corregido, de modo que no depende del modo que opera ahora.

Vea también **domdocument_get_elements_by_tagname()**

DomDocument->get_elements_by_tagname

(no version information, might be only in CVS)

DomDocument->get_elements_by_tagname --

Descripción

array **DomDocument->get_elements_by_tagname** (string nombre)

Vea también **domdocument_add_root()**

DomDocument->html_dump_mem

(no version information, might be only in CVS)

DomDocument->html_dump_mem -- Vuelca el árbol XML interno de vuelta a una cadena como HTML

Descripción

string **DomDocument->html_dump_mem** (void)

Crea un documento HTML para la representación dom. Por lo general, esta función es llamada después de construier un nuevo documento dom desde ceros, como en el siguiente ejemplo.

Ejemplo 1. Creación de una cabecera de documento HTML simple

```
<?php

// Crea el documento
$doc = domxml_new_doc("1.0");
$raiz = $doc->create_element("html");
$raiz = $doc->append_child($raiz);
$cabecera = $doc->create_element("head");
$cabecera = $raiz->append_child($cabecera);
$titulo = $doc->create_element("title");
$titulo = $cabecera->append_child($titulo);

$texto = $doc->create_text_node("This is the title");
$texto = $titulo->append_child($texto);

echo $doc->html_dump_mem();
?>
```

El resultado del ejemplo seria:

```
<html><head><title>This is the title</title></head></html>
```

Vea también **domdocument_dump_file()**, y **domdocument_html_dump_mem()**.

DomDocument->xinclude

(no version information, might be only in CVS)

DomDocument->xinclude -- Reemplaza sentencias XInclude en un Objeto DomDocument

Descripción

int **DomDocument->xinclude** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Esta función reemplaza elementos XInclude en un objeto DomDocument.

Ejemplo 1. Sustitución

```
<?php

// include.xml contiene :
// <child>prueba</child>

$xml = '<?xml version="1.0"?>
<root xmlns:xi="http://www.w3.org/2001/XInclude">
  <xi:include href="include.xml">
    <xi:fallback>
      <error>xinclude: no se encontr&oacute; include.xml</error>
    </xi:fallback>
  </xi:include>
</root>';

$domxml = domxml_open_mem($xml);
$domxml->xinclude();

echo $domxml->dump_mem();

?>
```

El resultado del ejemplo seria:

```
<?xml version="1.0"?>
<root xmlns:xi="http://www.w3.org/2001/XInclude">
  <child>prueba</child>
</root>
```

Si `include.xml` no existe, usted verá:

```
<?xml version="1.0"?>
<root xmlns:xi="http://www.w3.org/2001/XInclude">
  <error>xinclude: no se encontr&oacute; include.xml</error>
</root>
```

DomDocumentType->entities

(no version information, might be only in CVS)

DomDocumentType->entities -- Devuelve una lista de entidades

Descripción

array **DomDocumentType->entities** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

DomDocumentType->internal_subset

(no version information, might be only in CVS)

DomDocumentType->internal_subset -- Devuelve el sub-conjunto interno

Descripción

bool **DomDocumentType->internal_subset** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

DomDocumentType->name

(no version information, might be only in CVS)

DomDocumentType->name -- Devuelve el nombre del tipo de documento

Descripción

string **DomDocumentType->name** (void)

Esta función devuelve el nombre del tipo de documento.

Ejemplo 1. Obtener el nombre del tipo de documento

```
<?php

include("ejemplo.inc");

if (!$dom = domxml_open_mem($cadena_xml)) {
    echo "Ocurri&oacute; un error al interpretar el documento\n";
    exit;
}

$tipo_doc = $dom->doctype();
echo $tipo_doc->name(); // chapter

?>
```

DomDocumentType->notations

(no version information, might be only in CVS)

DomDocumentType->notations -- Devuelve una lista de notaciones

Descripción

array **DomDocumentType->notations** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

DomDocumentType->public_id

(no version information, might be only in CVS)

DomDocumentType->public_id -- Devuelve el id público del tipo de documento

Descripción

string **DomDocumentType->public_id** (void)

Esta función devuelve el id público del tipo de documento.

El siguiente ejemplo no produce ninguna salida.

Ejemplo 1. Recuperación del id público

```
<?php
include("ejemplo.inc");

if (!$dom = domxml_open_mem($cadena_xml)) {
  echo "Ocurri&oacute; un error al analizar el documento\n";
  exit;
}

$tipo_doc = $dom->doctype();
echo $tipo_doc->public_id();
?>
```

DomDocumentType->system_id

(no version information, might be only in CVS)

DomDocumentType->system_id -- Devuelve el id de sistema del tipo de documento

Descripción

string **DomDocumentType->system_id** (void)

Devuelve el id de sistema del tipo de documento.

Ejemplo 1. Recuperación del id de sistema

```
<?php
include("ejemplo.inc");

if (!$dom = domxml_open_mem($cadena_xml)) {
  echo "Ocurri&oacute; un error mientras se analizaba el documento\n";
  exit;
}

$tipo_doc = $dom->doctype();
echo $tipo_doc->system_id();
?>
```

El resultado del ejemplo seria:

```
/share/sgml/Norman_Walsh/db3xml10/db3xml10.dtd
```

DomElement->get_attribute_node

(no version information, might be only in CVS)

DomElement->get_attribute_node -- Devuelve el nodo del atributo dado

Descripción

object **DomElement->get_attribute_node** (string nombre)

Devuelve el nodo del atributo *nombre* en el elemento actual. El parámetro *nombre* es sensible a mayúsculas y minúsculas.

Si no se encuentra un atributo con el nombre dado, se devuelve **FALSE**.

Ejemplo 1. Obtener un nodo de atributo

```
<?php

include("ejemplo.inc");

if (!$dom = domxml_open_mem($cadena_xml)) {
    echo "Ocurri&oacute; un error al interpretar el documento\n";
    exit;
}

$raiz = $dom->document_element();
if ($atributo = $raiz->get_attribute_node('language')) {
    echo 'El idioma es: ' . $atributo->value() . "\n";
}

?>
```

DomElement->get_attribute

(no version information, might be only in CVS)

DomElement->get_attribute -- Devuelve el valor del atributo dado

Descripción

string **DomElement->get_attribute** (string nombre)

Devuelve el valor del atributo *nombre* en el nodo actual. El parámetro *nombre* es sensible a mayúsculas y minúsculas.

A partir de PHP 4.3, si no se encuentra un atributo con el *nombre* dado, se devuelve una cadena vacía.

Ejemplo 1. Obtener el valor de un atributo

```
<?php

include("ejemplo.inc");

if (!$dom = domxml_open_mem($cadena_xml)) {
    echo "Ocurri&oacute; un error al interpretar el documento\n";
    exit;
}

// obtener chapter
$raiz = $dom->document_element();
echo $raiz->get_attribute('language'); // en

?>
```

Vea también **domelement_set_attribute()**

DomElement->get_elements_by_tagname

(no version information, might be only in CVS)

DomElement->get_elements_by_tagname -- Obtiene elementos por el nombre de etiqueta

Descripción

array **DomElement->get_elements_by_tagname** (string nombre)

Esta función devuelve una matriz con todos los elementos que tienen *nombre* como su nombre de etiqueta. Cada elemento de la matriz es de tipo DomElement.

Ejemplo 1. Obtener el contenido

```
<?php
if (!$dom = domxml_open_mem($cadena_xml)) {
  echo "Ocurri&oacute; un error al analizar el documento\n";
  exit;
}

$raiz = $dom->document_element();

$matriz_nodos = $raiz->get_elements_by_tagname("element");

for ($i = 0; $i<count($matriz_nodos); $i++) {
    $nodo = $matriz_nodos[$i];
    echo "El elemento[$i] es: " . $nodo->get_content();
}

?>
```

DomElement->has_attribute

(no version information, might be only in CVS)

DomElement->has_attribute -- Verifica si un atributo existe en el nodo actual

Descripción

bool **DomElement->has_attribute** (string nombre)

Esta función verifica si un atributo con el *nombre* dado existe en el nodo actual.

Ejemplo 1. Probar la existencia de un atributo

```
<?php

include("ejemplo.inc");

if (!$dom = domxml_open_mem($cadena_xml)) {
    echo "Ocurri&oacute; un error al interpretar el documento\n";
    exit;
}

$raiz = $dom->document_element();

$bufer = '<html'
if ($raiz->has_attribute('language')) {
    $buffer .= 'lang="' . $raiz->get_attribute('language') . '"';
}
$bufer .= '>';

?>
```

DomElement->remove_attribute

(no version information, might be only in CVS)

DomElement->remove_attribute -- Elimina un atributo

Descripción

bool **DomElement->remove_attribute** (string nombre)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

DomElement->set_attribute

(no version information, might be only in CVS)

DomElement->set_attribute -- Define el valor de un atributo

Descripción

object **DomElement->set_attribute** (string nombre, string valor)

Define un antributo con el nombre *nombre* con el *valor* dado. Si el atributo no existe, será creado.

Ejemplo 1. Definición de un atributo

```
<?php
$doc = domxml_new_doc("1.0");
$nodo = $doc->create_element("para");
$nodo_nuevo = $doc->append_child($nodo);
$nodo_nuevo->set_attribute("align", "left");
?>
```

Vea también **domelement_get_attribute()**.

DomElement->tagname

(no version information, might be only in CVS)

DomElement->tagname -- Devuelve el nombre del elemento actual

Descripción

string **DomElement->tagname** (void)

Devuelve el nombre del nodo actual. Llamar esta función es ñalente a acceder a la propiedad *tagname*, o llamar **DomElement->node_name()** en el nodo actual.

Ejemplo 1. Obtener el nombre del nodo

```php
<?php

include("ejemplo.inc");

if (!$dom = domxml_open_mem($cadena_xml)) {
    echo "Ocurri&oacute; un error al interpretar el documento\n";
    exit;
}

$raiz = $dom->document_element();
echo $raiz->tagname();   // chapter
echo $raiz->tagname;     // chapter
echo $raiz->node_name(); // chapter


?>
```

DomNode->add_namespace

(no version information, might be only in CVS)

DomNode->add_namespace -- Agrega una declaración de espacio de nombres a un nodo

Descripción

bool **DomNode->add_namespace** (string uri, string prefijo)

Vea también **domdocument_create_element_ns()**, y **domnode_set_namespace()**

DomNode->append_child

(no version information, might be only in CVS)

DomNode->append_child -- Agrega un nuevo hijo al final del grupo de hijos

Descripción

object **DomNode->append_child** (object nodo_nuevo)

Esta función agrega un hijo a una lista existente de hijos o crea una nueva lista de hijos. El hijo puede ser creado, por ejemplo, con **domdocument_create_element()**, **domdocument_create_text()** etc. o simplemente mediante el uso de otro nodo.

(PHP < 4.3) Antes de que un nuevo hijo sea agregado, éste es duplicado. Por lo tanto el nuevo hijo es una nueva copia que puede ser modificada sin cambiar el nodo que fue pasado a esta función. Si el nodo pasado tiene hijos, ellos serán duplicados también, lo que facilita la duplicación de grandes segmentos de un documento XML. El valor de retorno es el hijo agregado. Si planea realizar modificaciones posteriores sobre el hijo agregado, debe usar el nodo devuelto.

(PHP 4.3.0/4.3.1) El nuevo hijo *nodo_nuevo* es enlazado primero desde su contexto actual, si ya se trata de un hijo de DomNode. Por lo tanto el nodo es movido y no una copia del nodo.

(PHP >= 4.3.2) El nuevo hijo *nodo_nuevo* es primera separado de su contexto actual, si éste ya existe en el árbol. Por lo tanto el nodo es movido y no copiado. Este debe ser su comportamiento de

acuerdo a las especificaciones del W3C. Si desea duplicar segmentos grandes de un documento XML, use DomNode->clone_node() antes de agregar.

El siguiente ejemplo agrega un nuevo nodo tipo elemento a un documento nuevo y define el atributo "align" como "left".

Ejemplo 1. Agregar un hijo

```
<?php
$doc = domxml_new_doc("1.0");
$nodo = $doc->create_element("para");
$nodo_nuevo = $doc->append_child($nodo);
$nodo_nuevo->set_attribute("align", "left");
?>
```

El anterior ejemplo pudo ser escrito también como:

Ejemplo 2. Agregar un hijo

```
<?php
$doc = domxml_new_doc("1.0");
$nodo = $doc->create_element("para");
$nodo->set_attribute("align", "left");
$nodo_nuevo = $doc->append_child($nodo);
?>
```

Un ejemplo más complejo se presenta a continuación. Primero busca cierto elemento, lo duplica incluyendo sus hijos y lo agrega como un hermano. Finalmente un nuevo atributo es agregado a uno de los hijos del nuevo hermano y el documento completo es volcado.

Ejemplo 3. Agregar un hijo

```
<?php
include("ejemplo.inc");

if (!$dom = domxml_open_mem($cadena_xml)) {
  echo "Ocurri&oacute; un error al analizar el documento\n";
  exit;
}

$elementos = $dom->get_elements_by_tagname("informaltable");
print_r($elementos);
$elemento = $elementos[0];

$padre = $elemento->parent_node();
$nodo_nuevo = $padre->append_child($elemento);
$hijos = $nodo_nuevo->children();
$atr = $hijos[1]->set_attribute("align", "left");

echo "<pre>";
$archivo_xml = $dom->dump_mem();
echo htmlentities($archivo_xml);
echo "</pre>";
?>
```

El ejemplo anterior pudo crearse también con **domnode_insert_before()** en lugar de **domnode_append_child()**.

Vea también **domnode_insert_before()**, y **domnode_clone_node()**.

DomNode->append_sibling

(no version information, might be only in CVS)

DomNode->append_sibling -- Agrega un nuevo hermano a un nodo

Descripción

object **DomNode->append_sibling** (object nodo_nuevo)

Esta función agrega un hermano a un nodo existente. El hijo puede ser creado, por ejemplo, con **domdocument_create_element()**, **domdocument_create_text()** etc. o simplemente mediante el uso de otro nodo.

Antes de que un nuevo hermano sea agregado, éste es duplicado. Por lo tanto, el nuevo hijo es una copia completamente nueva que puede modificarse sin cambiar el nodo que fue pasado a ésta función. Si el nodo pasado tiene hijos, ellos serán duplicados también, lo que facilita enormemente la duplicación de partes grandes de un documento XML. El valor de retorno es el hermano agregado. Si planea hacer modificaciones posteriores al hermano agregado, debe usar el nodo devuelto.

Esta función ha sido agregada para ofrecer el comportamiento de **domnode_append_child()**, del modo que funcionaba hasta PHP 4.2.

Vea también **domnode_append_before()**.

DomNode->attributes

(no version information, might be only in CVS)

DomNode->attributes -- Devuelve la lista de atributos

Descripción

array **DomNode->attributes** (void)

Esta función solo devuelve una matriz de atributos si el nodo es de tipo XML_ELEMENT_NODE.

(Sólo PHP >= 4.3) Si no se encuentran atributos, se devuelve NULL.

DomNode->child_nodes

(no version information, might be only in CVS)

DomNode->child_nodes -- Devuelve los hijos del nodo

Descripción

array **DomNode->child_nodes** (void)

Devuelve todos los hijos del nodo.

Vea también **domnode_next_sibling()**, y **domnode_previous_sibling()**.

DomNode->clone_node

(no version information, might be only in CVS)

DomNode->clone_node -- Clona un nodo

Descripción

object **DomNode->clone_node** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

DomNode->dump_node

(no version information, might be only in CVS)

DomNode->dump_node -- Vuelca un nodo único

Descripción

string **DomNode->dump_node** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

Vea también **domdocument_dump_mem()**.

DomNode->first_child

(no version information, might be only in CVS)

DomNode->first_child -- Devuelve el primer hijo del nodo

Descripción

object **DomNode->first_child** (void)

Devuelve el primer hijo del nodo.

(Sólo PHP >= 4.3) Si no se encuentra un hijo, se devuelve NULL.

Vea también **domnode_last_child()**, y **domnode_next_sibling()**, **domnode_previous_sibling()**.

DomNode->get_content

(no version information, might be only in CVS)

DomNode->get_content -- Obtiene el contenido del nodo

Descripción

string **DomNode->get_content** (void)

Esta función devuelve el contenido del nodo mismo.

Ejemplo 1. Obtener el contenido

```
<?php
if (!$dom = domxml_open_mem($codena_xml)) {
  echo "Ocurri&oacute; un error al analizar el documento\n";
  exit;
}

$raiz = $dom->document_element();

$matriz_nodos = $raiz->get_elements_by_tagname("element");

for ($i = 0; $i<count($matriz_nodos); $i++) {
    $nodo = $matriz_nodos[$i];
    echo "El elemento[$i] is: " . $nodo->get_content();
}

?>
```

DomNode->has_attributes

(no version information, might be only in CVS)

DomNode->has_attributes -- Verifica si un nodo tiene atributos

Descripción

bool **DomNode->has_attributes** (void)

Esta función verifica si el nodo tiene atributos.

Vea también **domnode_has_child_nodes()**.

DomNode->has_child_nodes

(no version information, might be only in CVS)

DomNode->has_child_nodes -- Verifica si el nodo tiene hijos

Descripción

bool **DomNode->has_child_nodes** (void)

Esta función verifica si el nodo tiene hijos.

Vea también **domnode_child_nodes()**.

DomNode->insert_before

(no version information, might be only in CVS)

DomNode->insert_before -- Inserta un nodo nuevo como hijo

Descripción

object **DomNode->insert_before** (object nodo_nuevo, object nodo_ref)

Esta función inserta el nuevo nodo, *nodo_nuevo*, justo antes del nodo *nodo_ref*. El valor de retorno es el nodo insertado. Si planea hacer modificaciones posteriores sobre el hijo agregado, debe usar el nodo devuelto.

(Sólo PHP >= 4.3) Si *nodo_nuevo* ya es parte de un documento, será primero desenlazado de su contexto actual. Si *nodo_ref* es NULL, entonces *nodo_nuevo* será insertado al final de la lista de hijos.

domnode_insert_before() es bastante similar a **domnode_append_child()** como muestra el siguiente ejemplo, el cual hace lo mismo que el ejemplo en **domnode_append_child()**.

Ejemplo 1. Agregar un hijo

```
<?php
include("ejemplo.inc");

if (!$dom = domxml_open_mem($cadena_xml)) {
  echo "Ocurri&oacute; un error al analizar el documento\n";
  exit;
}

$elementos = $dom->get_elements_by_tagname("informaltable");
print_r($elementos);
$elemento = $elementos[0];

$nodo_nuevo = $elemento->insert_before($elemento, $elemento);
$hijos = $nodo_nuevo->children();
$atr = $hijos[1]->set_attribute("align", "left");

echo "<pre>";
$archivo_xml = $dom->dump_mem();
echo htmlentities($archivo_xml);
echo "</pre>";
?>
```

Vea también **domnode_append_child()**.

DomNode->is_blank_node

(no version information, might be only in CVS)

DomNode->is_blank_node -- Verifica si el nodo está en blanco

Descripción

bool **DomNode->is_blank_node** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

DomNode->last_child

(no version information, might be only in CVS)

DomNode->last_child -- Devuelve el último hijo del nodo

Descripción

object **DomNode->last_child** (void)

Devuelve el último hijo del nodo.

(Sólo PHP >= 4.3) Si no se encuentra un último hijo, se devuelve NULL.

Vea también **domnode_first_child()**, y **domnode_next_sibling()**, **domnode_previous_sibling()**.

DomNode->next_sibling

(no version information, might be only in CVS)

DomNode->next_sibling -- Devuelve el siguiente hermano del nodo

Descripción

object **DomNode->next_sibling** (void)

Esta función devuelve el siguiente hermano del nodo actual. Si no hay un siguiente hermano, devuelve **`FALSE`** (< 4.3) o null (>= 4.3). Es posible usar esta función para iterar sobre los hijos de un nodo, como se muestra en el ejemplo a continuación.

Ejemplo 1. Iterar sobre los hijos

```
<?php
include("ejemplo.inc");

if (!$dom = domxml_open_mem($cadena_xml)) {
  echo "Ocurri&oacute; un error al analizar el documento\n";
  exit;
}

$elementos = $dom->get_elements_by_tagname("tbody");
$elemento = $elementos[0];
$hijo = $elemento->first_child();

while ($hijo) {
   print_r($hijo);
   $hijo = $hijo->next_sibling();
}
?>
```

Vea también **domnode_previous_sibling()**.

DomNode->node_name

(no version information, might be only in CVS)

DomNode->node_name -- Devuelve el nombre del nodo

Descripción

string **DomNode->node_name** (void)

Devuelve el nombre del nodo. El nombre tiene significados diferentes para los diferentes tipos de nodo, como se ilustra en la siguiente tabla.

Tabla 1. Significado de valores

Tipo	Significado
DomAttribute	valor del atributo
DomAttribute	
DomCDataSection	sección-#cdata
DomComment	#comment
DomDocument	#document
DomDocumentType	nombre del tipo de documento
DomElement	nombre de la etiqueta
DomEntity	nombre de la entidad
DomEntityReference	nombre de referencia a entidad
DomNotation	nombre de notación
DomProcessingInstruction	destino
DomText	#text

DomNode->node_type

(no version information, might be only in CVS)

DomNode->node_type -- Devuelve el tipo de nodo

Descripción

int **DomNode->node_type** (void)

Devuelve el tipo del nodo. Todos los tipos posibles son listados en la tabla de la introducción.

Ejemplo 1.

```
<?php

include 'ejemplo.inc';

$dom = domxml_open_mem($cadena_xml);

$capitulo = $dom->document_element();

// Veamos los elementos contenidos en capitulo
foreach($capitulo->child_nodes() as $nodo) {
  if ($nodo->node_type() == XML_ELEMENT_NODE) {
    echo $nodo->node_name() . "\n";
  }
}

?>
```

El anterior ejemplo producirá la salida:

```
title
para
```

DomNode->node_value

(no version information, might be only in CVS)

DomNode->node_value -- Devuelve el valor de un nodo

Descripción

string **DomNode->node_value** (void)

Devuelve el valor del nodo. El valor tiene significados diferentes para los diferentes tipos de nodos, como se ilustra en la siguiente tabla.

Tabla 1. Significado de valores

Tipo	Significado
DomAttribute	valor del atributo
DomAttribute	
DomCDataSection	contenido

Tipo	Significado
DomComment	contenido del comentario
DomDocument	null
DomDocumentType	null
DomElement	null
DomEntity	null
DomEntityReference	null
DomNotation	null
DomProcessingInstruction	contenido completo sin destino
DomText	contenido del texto

DomNode->owner_document

(no version information, might be only in CVS)

DomNode->owner_document -- Devuelve el documento al que este nodo pertenece

Descripción

object **DomNode->owner_document** (void)

Esta función devuelve el documento al que pertenece el nodo actual.

El siguiente ejemplo creará dos listas idénticas de hijos.
Ejemplo 1. Encontrar el documento de un nodo

```
<?php
$doc = domxml_new_doc("1.0");
$nodo = $doc->create_element("para");
$nodo = $doc->append_child($nodo);
$hijos = $doc->children();
print_r($hijos);

$doc2 = $nodo->owner_document();
$hijos = $doc2->children();
print_r($hijos);
?>
```

Vea también **domnode_insert_before()**.

DomNode->parent_node

(no version information, might be only in CVS)

DomNode->parent_node -- Devuelve el padre del nodo

Descripción

object **DomNode->parent_node** (void)

Esta función devuelve el nodo padre.

(Sólo PHP >= 4.3) Si no se encuentra un padre, se devuelve NULL.

El siguiente ejemplo mostrará dos listas idénticas de hijos.

Ejemplo 1. Encontrar el documento de un nodo

```
<?php
$doc = domxml_new_doc("1.0");
$nodo = $doc->create_element("para");
$nodo = $doc->append_child($nodo);
$hijos = $doc->children();
print_r($hijos);

$doc2 = $nodo->parent_node();
$hijos = $doc2->children();
print_r($hijos);
?>
```

DomNode->prefix

(no version information, might be only in CVS)

DomNode->prefix -- Devuelve el prefijo de espacio de nombres del nodo

Descripción

string **DomNode->prefix** (void)

Devuelve el prefijo de espacio de nombres del nodo.

DomNode->previous_sibling

(no version information, might be only in CVS)

DomNode->previous_sibling -- Devuelve el hermano anterior del nodo

Descripción

object **DomNode->previous_sibling** (void)

Esta función devuelve el hermano anterior del nodo actual. Si no hay un hermano anterior, devuelve **FALSE** (< 4.3) o **NULL** (>= 4.3). Es posible usar esta función para iterar a través de los hijos de un nodo, como se muestra en el ejemplo.

Vea también **domnode_next_sibling()**.

DomNode->remove_child

(no version information, might be only in CVS)

DomNode->remove_child -- Elimina un hijo de una lista de hijos

Descripción

object **DomNode->remove_child** (object hijo_antiguo)

Esta función elimina un hijo de una lista de hijos. Si el hijo no puede ser eliminado, o no es un hijo, la función devuelve **FALSE**. Si el hijo puede ser eliminado, la función devuelve el hijo antiguo.

Ejemplo 1. Eliminar un hijo

```
<?php
include("ejemplo.inc");

if (!$dom = domxml_open_mem($cadena_xml)) {
  echo "Ocurri&oacute; un error al analizar el documento\n";
  exit;
}

$elementos = $dom->get_elements_by_tagname("tbody");
$elemento = $elementos[0];
$hijos = $elemento->child_nodes();
$hijo = $elemento->remove_child($hijos[0]);

echo "<PRE>";
$archivo_xml = $dom->dump_mem(true);
echo htmlentities($archivo_xml);
echo "</PRE>";
?>
```

Vea también **domnode_append_child()**.

DomNode->replace_child

(no version information, might be only in CVS)

DomNode->replace_child -- Reemplaza un hijo

Descripción

object **DomNode->replace_child** (object nodo_antiguo, object nodo_nuevo)

(PHP 4.2) Esta función reemplaza el hijo *nodo_antiguo* con el nuevo nodo pasado. Si el nuevo nodo ya es un hijo, no será agregado una segunda vez. Si el nodo antiguo no puede encontrarse, la función devuelve **FALSE**. Si el reemplazo tiene éxito, se devuelve el nodo antiguo.

(PHP 4.3) Esta función reemplaza el hijo *nodo_antiguo* con el *nodo_nuevo* pasado, incluso si el nuevo nodo ya es un hijo del DomNode. Si *nodo_nuevo* ya había sido insertado en el documento, primero es desenlazado de su contexto actual. Si el nodo antiguo no puede ser encontrado, la función devuelve **FALSE**. Si el reemplazo tiene éxito, se devuelve el nodo antiguo. (Este comportamiento sigue las especificaciones del W3C).

Vea también **domnode_append_child()**

DomNode->replace_node

(no version information, might be only in CVS)

DomNode->replace_node -- Reemplaza el nodo

Descripción

object **DomNode->replace_node** (object nodo_nuevo)

(PHP 4.2) Esta función reemplaza un nodo existente con el nuevo nodo pasado. Antes del reemplazo, *nodo_nuevo* es copiado si tiene un padre, para asegurarse de que un nodo que ya esté en el documento no sea insertado una segunda vez. Este comportamiento obliga a hacer todas las modificaciones en el nodo antes del reemplazo, o retomar el nodo insertado después con funciones como **domnode_first_child()**, **domnode_child_nodes()** etc..

(PHP 4.3) Esta función reemplaza un nodo existente con el nuevo nodo pasado. No sigue siendo copiado. Si *nodo_nuevo* ya había sido insertado en el documento, es primero despojado del enlace con su contexto actual. Si el reemplazo tiene éxito, se devuelve el nodo antiguo.

Vea también **domnode_append_child()**

DomNode->set_content

(no version information, might be only in CVS)

DomNode->set_content -- Define el contenido del nodo

Descripción

bool **DomNode->set_content** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

DomNode->set_name

(no version information, might be only in CVS)

DomNode->set_name -- Define el nombre del nodo

Descripción

bool **DomNode->set_name** (void)

Define el nombre del nodo.

Vea también **domnode_node_name()**.

DomNode->set_namespace

(no version information, might be only in CVS)

DomNode->set_namespace -- Define el espacio de nombres de un nodo

Descripción

void **DomNode->set_namespace** (string uri [, string prefijo])

Define el espacio de nombres de un nodo a *uri*. Si ya existe una declaración de espacio de nombres con el mismo uri en uno de los nodos padres del nodo, tal prefijo será tomado, de lo contrario tomará aquél indicado en el parámetro opcional *prefijo* o generará uno aleatorio.

Vea también **domdocument_create_element_ns()**, y **domnode_add_namespace()**

DomNode->unlink_node

(no version information, might be only in CVS)

DomNode->unlink_node -- Elimina el nodo

Descripción

object **DomNode->unlink_node** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

DomProcessingInstruction->data

(no version information, might be only in CVS)

DomProcessingInstruction->data -- Devuelve los datos de un nodo pi

Descripción

string **DomProcessingInstruction->data** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

DomProcessingInstruction->target

(no version information, might be only in CVS)

DomProcessingInstruction->target -- Devuelve el destino de un nodo pi

Descripción

string **DomProcessingInstruction->target** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

DomXsltStylesheet->process

(no version information, might be only in CVS)

DomXsltStylesheet->process -- Aplica la Transformación XSLT sobre un objeto DomDocument

Descripción

object **DomXsltStylesheet->process** (object DomDocument [, array parametros_xslt [, bool param_es_xpath]])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

Vea también **domxml_xslt_stylesheet()**, **domxml_xslt_stylesheet_file()**, y **domxml_xslt_stylesheet_doc()**

DomXsltStylesheet->result_dump_file

(no version information, might be only in CVS)

DomXsltStylesheet->result_dump_file -- Vuelca el resultado de una Transformación XSLT a un archivo

Descripción

string **DomXsltStylesheet->result_dump_file** (object DomDocument, string nombre_archivo)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Esta función sólo se encuentra disponible a partir de PHP 4.3

Dado que DomXsltStylesheet->process() siempre devuelve un DomDocument XML bien formado, sin importar el método de salida declarado en <xsl:output> y atributos o elementos similares, no es de mucha utilidad si desea generar datos HTML 4 o de texto. Por otra parte, esta función honra <xsl:output method="html|text"> y otras directivas de control de salida. Vea el ejemplo para más

información sobre su uso.

Ejemplo 1. Guardar el resultado de una transformción XSLT en un archivo

```
<?php
$nombre_archivo = "stylesheet.xsl";
$doc_xml = domxml_open_file("datos.xml");
$doc_xsl = domxml_xslt_stylesheet_file($nombre_archivo);
$resultado = $doc_xsl->process($doc_xml);
echo $doc_xsl->result_dump_file($resultado, "nombre_archivo");
?>
```

Vea también **domxml_xslt_result_dump_mem()**, y **domxml_xslt_process()**

DomXsltStylesheet->result_dump_mem

(no version information, might be only in CVS)

DomXsltStylesheet->result_dump_mem -- Vuelca el resultado de una Transformación XSLT de vuelta a una cadena

Descripción

string **DomXsltStylesheet->result_dump_mem** (object DomDocument)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Esta función sólo se encuentra disponible a partir de PHP 4.3

Dado que DomXsltStylesheet->process() siempre devuelve un DomDocument XML bien formado, sin importar el método de salida declarado en <xsl:output> y atributos o elementos similares, no es de mucha utilidad si desea generar datos HTML 4 o de texto. Por otra parte, esta función honra <xsl:output method="html|text"> y otras directivas de control de salida. Vea el ejemplo para más información sobre su uso.

Ejemplo 1. Imprimir el resultado de una transformación XSLT

```
<?php
$nombre_archivo = "stylesheet.xsl";
$doc_xml = domxml_open_file("datos.xml");
$doc_xsl = domxml_xslt_stylesheet_file($nombre_archivo);
$resultado = $doc_xsl->process($doc_xml);
echo $doc_xsl->result_dump_mem($resultado);
?>
```

Vea también **domxml_xslt_result_dump_file()**, y **domxml_xslt_process()**

domxml_new_doc

(PHP 4 >= 4.2.1)

domxml_new_doc -- Crea un nuevo documento XML vacío

Descripción

object **domxml_new_doc** (string version)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Crea un nuevo documento dom desde ceros y lo devuelve.

Vea también **domdocument_add_root()**

domxml_open_file

(PHP 4 >= 4.2.1)

domxml_open_file -- Crea un objeto DOM a partir de un archivo XML

Descripción

object **domxml_open_file** (string nombre_archivo [, int modo [, array &error]])

Esta función analiza el documento XML en el archivo con nombre *nombre_archivo* y devuelve un objeto de clase "documento Dom", que tiene las propiedades listadas anteriormente. El archivo es abierto en modo de sólo-lectura.

El parámetro opcional *modo* puede ser usado para modificar el comportamiento de ésta función. Fue agregado en PHP 4.3.0. Puede usar una de las siguientes constantes para este valor: **DOMXML_LOAD_PARSING** (predeterminado), **DOMXML_LOAD_VALIDATING** o **DOMXML_LOAD_RECOVERING**. También puede agregar **DOMXML_LOAD_DONT_KEEP_BLANKS**, **DOMXML_LOAD_SUBSTITUTE_ENTITIES** y **DOMXML_LOAD_COMPLETE_ATTRS** con una operación OR de bits.

Si se usa el parámetro *error*, éste contendrá los mensajes de error. *error* debe ser pasado por referencia. El parámetro fue agregado en PHP 4.3.0.

Ejemplo 1. Abrir un documento XML desde un archivo

```
<?php

if (!$dom = domxml_open_file("ejemplo.xml")) {
  echo "Ocurri&oacute; un error al analizar el documento\n";
  exit;
}

$raiz = $dom->document_element();
?>
```

Vea también **domxml_open_mem()**, y **domxml_new_doc()**.

domxml_open_mem

(PHP 4 >= 4.2.1)

domxml_open_mem -- Crea un objeto DOM desde un documento XML

Descripción

object **domxml_open_mem** (string cadena [, int mode [, array &error]])

Esta función analiza el documento XML en *cadena* y devuelve un objeto de clase "documento Dom", que tiene las propiedades listadas anteriormente. Esta función, **domxml_open_file()** o **domxml_new_doc()** debe llamarse antes de cualquier otra función.

El parámetro opcional *modo* puede ser usado para modificar el comportamiento de ésta función. Fue agregado en PHP 4.3.0. Vea **domxml_open_file()** para los posibles valores.

Si el parámetro *error* es usado, éste contendrá los mensajes de error. *error* debe ser pasado por referencia. El parámetro fue agregado en PHP 4.3.0.

Ejemplo 1. Abrir un documento XML desde una cadena

```
<?php
include("ejemplo.inc");

if (!$dom = domxml_open_mem($cadena_xml)) {
  echo "Ocurri&oacute;n un error al analizar el documento\n";
  exit;
}

$raiz = $dom->document_element();
?>
```

Vea también **domxml_open_file()**, y **domxml_new_doc()**.

domxml_version

(PHP 4 >= 4.1.0)

domxml_version -- Obtiene la versión de la biblioteca XML

Descripción

string **domxml_version** (void)

Esta función devuelve la versión de la biblioteca XML usada actualmente.

Ejemplo 1. Ejemplo de domxml_version()

```
<?php

echo domxml_version();

?>
```

El resultado del ejemplo seria algo similar a:

```
20607
```

domxml_xmltree

(PHP 4 >= 4.2.1)

domxml_xmltree -- Crea un árbol de objetos PHP a partir de un documento XML

Descripción

object **domxml_xmltree** (string cadena)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

La función analiza el documento XML en *cadena* y devuelve un árbol de objetos PHP como el documento interpretado. Esta función se encuentra separada de las demás, lo que quiere decir que no puede acceder al árbol con cualquiera de las otras funciones. Modificarlo, por ejemplo agregando nodos, no tiene sentido ya que no hay forma de volcar el resultado como un archivo XML. Sin embargo, esta función puede resultar útil si quiere leer un archivo y consultar su contenido.

domxml_xslt_stylesheet_doc

(PHP 4 >= 4.2.0)

domxml_xslt_stylesheet_doc -- Crea un Objeto DomXsltStylesheet a partir de un Objeto DomDocument

Descripción

object **domxml_xslt_stylesheet_doc** (object DocDocument)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.
Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

Vea también **domxsltstylesheet->process()**, **domxml_xslt_stylesheet()**, y **domxml_xslt_stylesheet_file()**.

domxml_xslt_stylesheet_file

(PHP 4 >= 4.2.0)

domxml_xslt_stylesheet_file -- Crea un Objeto DomXsltStylesheet a partir de un documento XSL en un archivo

Descripción

object **domxml_xslt_stylesheet_file** (string archivo_xsl)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

Vea también **domxsltstylesheet->process()**, **domxml_xslt_stylesheet()**, y **domxml_xslt_stylesheet_doc()**

domxml_xslt_stylesheet

(PHP 4 >= 4.2.0)

domxml_xslt_stylesheet -- Crea un Objeto DomXsltStylesheet desde un documento XML en una cadena

Descripción

object **domxml_xslt_stylesheet** (string documento_xsl)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

Vea también **domxsltstylesheet->process()**, **domxml_xslt_stylesheet_file()**, y **domxml_xslt_stylesheet_doc()**

xpath_eval_expression

(PHP 4 >= 4.0.4)

xpath_eval_expression -- Evalúa la Ruta de Ubicación XPath en la cadena entregada

Descripción

object **xpath_eval_expression** (object contexto_xpath, string expresion)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Ejemplo 1. Ejemplo de xpath_eval_expression()

```
<?php

include("ejemplo.inc");

if (!$dom = domxml_open_mem($cadena_xml)) {
    echo "Ocurri&oacute; un error al interpretar el documento\n";
    exit;
}

$xpath = xpath_new_context($dom);
var_dump(xpath_eval_expression($xpath, '/chapter/@language'));

?>
```

El resultado del ejemplo seria:

```
object(XPathObject)(2) {
   ["type"]=>
   int(1)
   ["nodeset"]=>
   array(1) {
     [0]=>
     object(domattribute)(5) {
       ["type"]=>
       int(2)
       ["name"]=>
       string(8) "language"
       ["value"]=>
       string(2) "en"
       [0]=>
       int(7)
       [1]=>
       int(138004256)
     }
  }
}
```

Vea también **xpath_eval()**.

xpath_eval

(PHP 4 >= 4.0.4)

xpath_eval -- Evalúa la Ruta de Ubicación XPatch en la cadena dada

Descripción

array **xpath_eval** (object contexto_xpath, string expresion_xpath [, object nodo_contexto])

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

El valor opcional *nodo_contexto* puede especificarse para realizar consultas XPath relativas.

Vea también **xpath_new_context()**.

xpath_new_context

(PHP 4 >= 4.0.4)

xpath_new_context -- Crea un nuevo contexto xpath

Descripción

object **xpath_new_context** (object documento_dom)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Vea también **xpath_eval()**.

xptr_eval

(PHP 4 >= 4.0.4)

xptr_eval -- Evalúa la Ruta de Ubicación XPtr en la cadena dada

Descripción

int **xptr_eval** ([object contexto_xpath, string cadena_eval])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

xptr_new_context

(PHP 4 >= 4.0.4)

xptr_new_context -- Crea un nuevo Contexto XPath

Descripción

string **xptr_new_context** ([object gestor_doc])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

XXIX. .NET Functions

Introducción

Aviso
Esta extensión es *EXPERIMENTAL*. Esto significa que el comportamiento de esta extensión, los nombre de sus funciones y en definitiva TODO lo documentado sobre esta extensión, puede cambiar en una futura versión de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Tabla de contenidos

dotnet_load

(no version information, might be only in CVS)

dotnet_load -- Loads a DOTNET module

Description

int **dotnet_load** (string assembly_name [, string datatype_name [, int codepage]])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

XXX. Funciones de Gestión de Errores y Registros

Introducción

Estas son funciones que trabajan en la gestión de errores y registros. Le permiten definir sus propias

reglas de gestión de errores, asi como modificar el modo en que los errores son registrados. Esto le permite modificar y mejorar el reporte de errores para que se adapte a sus necesidades.

Con las funciones de registro, puede enviar mensajes directamente a otras máquinas, a un correo electrónico (¡o correo electrónico a una puerta de enlace con un buscapersonas!), a bitácoras del sistema, etc., de modo que puede registrar y monitorear selectivamente las partes más importantes de sus aplicaciones y sitios web.

Las funciones de reporte de errores le permiten personalizar el nivel y tipo de retroalimentación de errores que PHP entrega, desde noticias simples a funciones personalizadas devueltas durante los errores.

Requirimientos

No se necesitan bibliotecas externas para construir esta extensión

---

Instalación

No se necesita ninguna instalación para usar estas funciones, son parte del núcleo de PHP.

Configuración en tiempo de ejecución

El comportamiento de estas funciones está afectado por los valores definidos en `php.ini`.

Tabla 1. Opciones de Configuración de Errores y Registro

Nombre	Por defecto	Modificable
error_reporting	E_ALL & ~E_NOTICE	PHP_INI_ALL
display_errors	"1"	PHP_INI_ALL
display_startup_errors	"0"	PHP_INI_ALL
log_errors	"0"	PHP_INI_ALL
log_errors_max_len	"1024"	PHP_INI_ALL
ignore_repeated_errors	"0"	PHP_INI_ALL
ignore_repeated_source	"0"	PHP_INI_ALL
report_memleaks	"1"	PHP_INI_ALL
track_errors	"0"	PHP_INI_ALL
html_errors	"1"	PHP_INI_ALL
docref_root	""	PHP_INI_ALL
docref_ext	""	PHP_INI_ALL

Nombre	Por defecto	Modificable
error_prepend_string	NULL	PHP_INI_ALL
error_append_string	NULL	PHP_INI_ALL
error_log	NULL	PHP_INI_ALL
warn_plus_overloading	NULL	PHP_INI??

Para más detalles y la definición de las constantes PHP_INI_* consulte **ini_set()**.

A continuación se presenta una corta explicación de las directivas de configuración.

error_reporting **integer**

Establece el nivel de reporte de errores. Este parámetro es, o bien un entero que representa un campo de bit, o constantes con nombre. Los niveles de error_reporting y las constantes están descritas en Constantes Predefinidas, y en `php.ini`. Para definir este valor en tiempo de ejecución, use la función **error_reporting()**. Vea también la directiva display_errors.

En PHP 4 y PHP 5, el valor predeterminado es E_ALL & ~E_NOTICE. Este valor no muestra errores de nivel **`E_NOTICE`**. Puede que usted quiera mostrarlos durante su actividad de desarrollo.

Nota: Al habilitar **`E_NOTICE`** durante el desarrollo de software tiene algunos beneficios. Para propósitos de depuración: los mensajes NOTICE le advertirán sobre posibles fallos en su código. Por ejemplo, se le advertirá sobre el uso de valores no-asignados. Es extremadamente útil para encontrar errores ortográficos y ahorrar tiempo de depuración. Los mensajes NOTICE le advertirán sobre el uso de un estilo incorrecto. Por ejemplo, $matriz[item] está mejor escrito como $matriz['item'] ya que PHP intenta darle a "item" un tratamiento de constante. Si no es una constante, PHP asume que es un índice de tipo cadena para la matriz.

Nota: En PHP 5, un nuevo nivel de error, **`E_STRICT`**, se encuentra disponible. Dado que **`E_STRICT`** no se incluye en **`E_ALL`**, usted debe habilitar explícitamente este tipo de nivel de error. Habilitar **`E_STRICT`** durante el desarrollo tiene algunos beneficios. Los mensajes STRICT le ayudarán a usar los últimos y más grandiosos métodos sugeridos para escribir código, por ejemplo, le advertirá sobre el uso de funciones obsoletas.

En PHP 3, el valor por defecto es *(E_ERROR | E_WARNING | E_PARSE)*, lo que en la práctica representa lo mismo. Note, sin embargo, que dado que las constantes en el archivo `php3.ini` de PHP 3 no son soportadas, el valor de error_reporting allí debe ser numérico; por lo tanto, es *7*.

display_errors **boolean**

Este parámetro determina si los errores deben ser puestos en pantalla como parte de la salida o si deben ser ocultados al usuario.

Nota: Es una característica de apoyo para su proceso de desarrollo y no debería ser usada nunca en sistemas en producción (p.ej. sistemas conectados a internet).

display_startup_errors **boolean**

Aun cuando display_errors esté encendido, los errores que ocurren durante la secuencia de arranque de PHP no son desplegados. Es muy recomendable mantener display_startup_errors apagado, excepto en tiempos de depuración.

log_errors **boolean**

Indica si los mensajes de error de un script deben ser registrados en la bitácora de errores del servidor o error_log. Esta opción es por lo tanto espocífica al servidor.

Nota: Es altamente recomendable que use registro de errores en lugar de despliegue de errores en sitios web en producción.

log_errors_max_len **integer**

Establece la longitud máxima de log_errors en bytes. En error_log se añade información sobre la fuente. El valor por defecto es 1024, y un valor de 0 permite que no se aplique ninguna longitud máxima en absoluto. Esta longitud se aplica también a los errores almacenados en el registro, a los errores que se despliegan y a $php_errormsg.

Cuando se usa un número entero, el valor del mismo es medido en bytes. También se puede usar la notación reducida tal como se describe en esta FAQ.

ignore_repeated_errors **boolean**

No registrar mensajes repetidos. Los errores repetidos deben ocurrir en el mismo archivo, en la misma línea hasta que el parámetro ignore_repeated_source sea establecido como true.

ignore_repeated_source **boolean**

Ignorar la fuente de mensaje cuando se ignoran los mensajes repetidos. Cuando este parámetro está encendido, no se registrará errores con mensajes repetidos desde diferentes fuentes de archivos.

report_memleaks **boolean**

Si este parámetro está apagado, entonces las fugas de memoria no serán mostradas (en stdout o en el registro). Este valor sólo tiene efecto en una compilación de depuración, y si error_reporting incluye E_WARNING en la lista permitida.

track_errors **boolean**

Cuando está habilitado, el último mensaje de error estará siempre presente en la variable $php_errormsg.

html_errors **boolean**

Permite deshabilitar las etiquetas HTML en los mensajes de error. El nuevo formato para los errores en HTML produce mensajes con enlaces que dirigen al usuario a una página que describe el error o función que ha causado el error. Estas referencias son afectadas por docref_root y docref_ext.

docref_root **string**

El nuevo formato de error contiene una referencia hacia una página que describe el error o la

función que ha causado el error. En el caso de páginas de un manual, usted puede descargar el manual en su idioma y establecer esta directiva ini como la URL de su copia local. Si su copia local del manual puede encontrarse bajo '/manual/' entonces puede usar simplemente `docref_root=/manual/`. Adicionalmente debe establecer un valor de docref_ext que coincida con las extensiones de archivo de su copia; `docref_ext=.html`. Es posible usar referencias externas. Por ejemplo, puede usar `docref_root=http://manual/en/` o `docref_root="http://landonize.it/?how=url&theme=classic&filter=Landon &url=http%3A%2F%2Fwww.php.net%2F"`.

Por lo general usted querrá que el valor de docref_root finalice con una barra '/'. Pero observe el segundo ejemplo mencionado anteriormente, el cual no tiene una barra final, ni la necesita.

> **Nota:** Esta es una característica de apoyo para su desarrollo ya que permite consultar la descripción de una función fácilmente. Sin embargo, nunca debe ser utilizada en sistemas en producción (p.ej. sistema conectados a internet).

docref_ext **string**

Vea docref_root.

> **Nota:** El valor de docref_ext debe comenzar con un punto '.'.

error_prepend_string **string**

Cadena a mostrar antes de un mensaje de error.

error_append_string **string**

Cadena a mostrar después de un mensaje de error.

error_log **string**

Nombre del archivo en donde deberían registrarse los errores del script. Si se utiliza el valor especial *syslog*, los errores son enviados al gestor de registros del sistema. En Unix, esto quiere decir syslog(3) y en Windows NT quiere decir el "event log". El gestor de registro de actividades no está soportado bajo Windows 95. Vea también: **syslog()**.

warn_plus_overloading **boolean**

Si se habilita, esta opción hace que PHP genere una advertencia cuando se utilice un operador más (+) con cadenas. Esto facilita la labor de encontrar scripts que necesitan ser reescritos para hacer uso del concatenador de cadenas en su lugar (.).

Constantes predefinidas

Las constantes listadas aqui están siempre disponibles a través del "núcleo PHP".

> **Nota:** Es posible usar estos nombres de constantes en `php.ini` pero no por fuera de PHP, como en `httpd.conf`, en donde debería usar los valores de máscara de bits en su lugar.

Tabla 2. Errores y Registro

Valor	Constante	Descripción	Nota
1	**E_ERROR** (integer)	Errores fatales en tiempo de ejecución. Estos indican errores de los que no es posible recuperarse, tales como problemas de asignación de memoria. Se detiene la ejecución del script.	
2	**E_WARNING** (integer)	Advertencias en tiempo de ejecución (errores no-fatales). La ejecución del script no se interrumpe.	
4	**E_PARSE** (integer)	Errores de intérprete en tiempo de compilación. Esto tipo de errores deberían ser generados únicamente por el interprete.	
8	**E_NOTICE** (integer)	Anotaciones en tiempo de ejecución. Indican que el script se ha topado con algo que puede indicar la presencia de un error, pero que también podría ocurrir en el curso normal de la ejecución de un script.	
16	**E_CORE_ERROR** (integer)	Errores fatales que ocurren durante el arranque inicial de PHP. Es como un **E_ERROR**, excepto que es generado por el núcleo de PHP.	a partir de PHP 4
32	**E_CORE_WARNING** (integer)	Advertencias (errores no-fatales) que ocurren durante el arranque inicial de PHP. Es como un **E_WARNING**, excepto que es generado por el núcleo de PHP.	a partir de PHP 4
64	**E_COMPILE_ERROR** (integer)	Errores fatales en tiempo de compilación. Es como un **E_ERROR**, excepto que es generado por el Motor de Scripting de Zend.	a partir de PHP 4
128	**E_COMPILE_WARNING** (integer)	Advertencias en tiempo de compilación (errores no fatales). Es como un **E_WARNING**, excepto que es generado por el Motor de Scripting de Zend.	a partir de PHP 4
256	**E_USER_ERROR** (integer)	Mensaje de error generado por el usuario. Es como un **E_ERROR**, excepto que es generado desde código PHP usando la función **trigger_error()**.	a partir de PHP 4
512	**E_USER_WARNING** (integer)	Mensaje de advertencia generado por el usuario. Es como un **E_WARNING**, excepto que es generado desde código PHP usando la función **trigger_error()**.	a partir de PHP 4
1024	**E_USER_NOTICE** (integer)	Anotación generada por el usuario. Es como un **E_NOTICE**, excepto que es generado desde código PHP usando la función **trigger_error()**.	a partir de PHP 4
2047	**E_ALL** (integer)	Todos los errores y advertencias, en la medida en que sean soportados, excepto por el nivel **E_STRICT**.	
2048	**E_STRICT** (integer)	Noticias de tiempo de ejecución. Habilite este valor para hacer que PHP sugiera cambios en su código que velarán por la mejor interoperabilidad y por mantener la compatibilidad de su código.	a partir de PHP 5

Los valores referidos anteriormente (ya sean numéricos o simbólicos) son usados para construir una máscara de bits que indica cuáles errores reportar. Puede usar los operadores bit a bit para combinar estos valores o excluir explícitamente ciertos tipos de errores. Sin embargo, note que únicamente '|', '~', '!', '^' y '&' serán entendidos desde `php.ini`, y que ningún operador bit a bit será interpretado correctamente desde `php3.ini`.

Ejemplos

A continuación podemos apreciar un ejemplo del uso de las capacidades de gestión de errores que vienen con PHP. Definimos una función de manipulación de errores que registra la información en un archivo (usando un formato XML), y envía un correo electrónico al desarrollador en caso de que un error crítico en la lógica del software ocurra.

Ejemplo 1. Uso de gestión de errores en un script

```php
<?php
// haremos nuestra propia manipulaci&oacute;n de errores
error_reporting(0);

// funcion de gestion de errores definida por el usuario
function gestorDeErroresDeUsuario($num_err, $mens_err, $nombre_archivo,
                                  $num_linea, $vars)
{

    // marca de fecha/hora para el registro de error
    $dt = date("Y-m-d H:i:s (T)");

    // definir una matriz asociativa de cadenas de error
    // en realidad las unicas entradas que deberiamos
    // considerar son E_WARNING, E_NOTICE, E_USER_ERROR,
    // E_USER_WARNING y E_USER_NOTICE

    $tipo_error = array (
                E_ERROR           => "Error",
                E_WARNING         => "Advertencia",
                E_PARSE           => "Error de Int&eacute;rprete",
                E_NOTICE          => "Anotaci&oacute;n",
                E_CORE_ERROR      => "Error de N&uacute;cleo",
                E_CORE_WARNING    => "Advertencia de N&uacute;cleo",
                E_COMPILE_ERROR   => "Error de Compilaci&oacute;n",
                E_COMPILE_WARNING => "Advertencia de Compilaci&oacute;n",
                E_USER_ERROR      => "Error de Usuario",
                E_USER_WARNING    => "Advertencia de Usuario",
                E_USER_NOTICE     => "Anotaci&oacute;n de Usuario",
                E_STRICT          => "Anotaci&oacute;n de tiempo de ejecuci&oacute;n"
                );
    // conjunto de errores de los cuales se almacenara un rastreo
    $errores_de_usuario = array(E_USER_ERROR, E_USER_WARNING, E_USER_NOTICE);

    $err = "<errorentry>\n";
    $err .= "\t<datetime>" . $dt . "</datetime>\n";
    $err .= "\t<errornum>" . $num_err . "</errornum>\n";
    $err .= "\t<errortype>" . $tipo_error[$num_err] . "</errortype>\n";
    $err .= "\t<errormsg>" . $mens_err . "</errormsg>\n";
    $err .= "\t<scriptname>" . $nombre_archivo . "</scriptname>\n";
    $err .= "\t<scriptlinenum>" . $num_linea . "</scriptlinenum>\n";

    if (in_array($num_err, $errores_de_usuario)) {
        $err .= "\t<vartrace>" . wddx_serialize_value($vars, "Variables") . "</vartrace
    }
    $err .= "</errorentry>\n\n";

    // para efectos de debug
    // echo $err;

    // guardar en el registro de errores, y enviar un correo
    // electr&oacute;nico si hay un error cr&iacute;tico de usuario
    error_log($err, 3, "/usr/local/php4/error.log");
    if ($num_err == E_USER_ERROR) {
        mail("phpdev@example.com", "Error Cr&iacute;tico de Usuario", $err);
    }
}


function distancia($vect1, $vect2)
{
    if (!is_array($vect1) || !is_array($vect2)) {
        trigger_error("Par&aacute;metros incorrectos, se esperan matrices", E_USER_ERRO
        return NULL;
    }

    if (count($vect1) != count($vect2)) {
        trigger_error("Los vectores deben ser del mismo tama&ntilde;o", E_USER_ERROR);
        return NULL;
    }

    for ($i=0; $i<count($vect1); $i++) {
```

Ver también

Vea también **syslog()**.

Tabla de contenidos

debug_backtrace

(PHP 4 >= 4.3.0, PHP 5)

debug_backtrace -- Genera un backtrace

Descripción

array **debug_backtrace** (void)

debug_backtrace() genera un backtrace PHP y devuelve esta información como un **array** asociativo. Los elementos posiblemente devueltos son referenciados en la siguiente tabla:

Tabla 1. Elementos posiblemente devueltos por debug_backtrace()

Nombre	Tipo	Descripción
function	**string**	El nombre de la función actual. Vea también __FUNCTION__.
line	**integer**	El número de línea actual. Vea también __LINE__.
file	**string**	El nombre del archivo actual. Vea también __FILE__.
class	**string**	El nombre de la clase actual. Vea también __CLASS__
type	**string**	El tipo de llamada actual. Si es una llamada de método, se devuelve "->". Si es una llamada a un método estático, se devuelve "::". Si es una llamada de función, no se devuelve nada.
args	**array**	Si se encuentra al interior de una función, contiene una lista de los argumentos de la función. Si se encuentra al interior de un archivo de inclusión, contiene una lista de los nombres de archivos incluídos.

El siguiente es un ejemplo sencillo.

Ejemplo 1. Ejemplo de debug_backtrace()

```php
<?php
// nombre de archivo: a.php

function una_prueba($cadena)
{
    echo "\nHola: $cadena";
    var_dump(debug_backtrace());
}

una_prueba('amigo');
?>

<?php
// nombre de archivo: b.php
include_once '/tmp/a.php';
?>
```

Resultados de ejecutar `/tmp/b.php`:

```
Hola: amigo
array(2) {
  [0]=>
  array(4) {
    ["file"] => string(10) "/tmp/a.php"
    ["line"] => int(10)
    ["function"] => string(8) "prueba_a"
    ["args"]=>
    array(1) {
      [0] => &string(5) "amigo"
    }
  }
  [1]=>
  array(4) {
    ["file"] => string(10) "/tmp/b.php"
    ["line"] => int(2)
    ["args"] =>
    array(1) {
      [0] => string(10) "/tmp/a.php"
    }
    ["function"] => string(12) "include_once"
  }
}
```

Vea también **trigger_error()** y **debug_print_backtrace()**.

debug_print_backtrace

(PHP 5)

debug_print_backtrace -- Imprime un backtrace

Descripción

void **debug_print_backtrace** (void)

debug_print_backtrace() imprime un backtrace PHP.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

Vea también **debug_backtrace()**.

error_log

(PHP 3, PHP 4 , PHP 5)

error_log -- Envía un mensaje de error a algún lugar

Descripción

int **error_log** (string message [, int message_type [, string destination [, string extra_headers]]])

Envía un mensaje de error al log de errores del servidor web, a un puerto TCP o a un fichero. El primer parámetro, *message* (mensaje), es el mensaje de error que debe ser registrado. El segundo parámetro, *message_type* (tipo de mensaje) indica el lugar al que debe dirigirse:

Tabla 1. error_log() tipos de log

0	*message* es enviado al registro de sistema de PHP, utilizando para ello el mecanismo de registro del Sistema Operativo (system logger) o un fichero, dependiendo del valor de la directiva de configuración error_log. Por defecto, esta es la opción que se utiliza.
1	*message* es enviado por correo electrónico a la dirección del parámetro *destination* (destino). Este es el único tipo de mensaje donde se utiliza el cuarto parámetro, *extra_headers*. Este tipo de mensaje utiliza la misma funcion que emplea **mail()**.
2	*message* es enviado a través de la conexión de depuración de PHP. Por tanto, esta opción está disponible sólo si la depuración remota ha sido activada. En este caso el parámetro *destination* especifica el nombre de host o dirección IP y, opcionalmente, el número de puerto del socket que recibe la información de depuración.
3	*message* es añadido al fichero *destination*.

Nota: Cuando se establece de forma explícita un valor de *3* al parámetro *message_type*, no se añade un carácter de nueva línea al final de la cadena *message*.

Aviso
La depuración remota a través de TCP/IP es una característica propia de PHP 3 que *no* está disponible en PHP 4.

Ejemplo 1. Ejemplos de uso de error_log()

```
<?php
// Enviar una notificaci&oacute;n al log del servidor si no se puede
// conectar a la base de datos.
if (!Ora_Logon($username, $password)) {
    error_log("Oracle database not available!", 0);
}

// Ejemplo de notificaci&oacute;n mediante correo electr&oacute;nico
if (!($foo = allocate_new_foo())) {
    error_log("Big trouble, we're all out of FOOs!", 1,
              "operator@example.com");
}

// Formas de uso alternativas de error_log():
error_log("You messed up!", 2, "127.0.0.1:7000");
error_log("You messed up!", 2, "loghost");
error_log("You messed up!", 3, "/var/tmp/my-errors.log");
?>
```

error_reporting

(PHP 3, PHP 4 , PHP 5)

error_reporting -- Establece que errores de PHP son informados

Descripción

int **error_reporting** ([int level])

La función **error_reporting()** permite establecer la directiva error_reporting en tiempo de ejecución. PHP tiene numerosos niveles de errores, que se pueden seleccionar de manera temporal (mientras dure la ejecución del script) mediante esta función.

error_reporting() establece el nivel de errores que notifica PHP y devuelve el nivel anterior. El parámetro *level* se puede establecer mediante una máscara de bits o mediante algunas constantes específicas. Se recomienda utilizar las constantes para asegurar la compatibilidad con versiones futuras. La causa de esta posible incompatibilidad es que a medida que se crean nuevos niveles de errores, aumenta el rango de números utilizado para indicarlo, por lo que podrían dejar de funcionar los anteriores niveles de errores indicados numéricamente.

Ejemplo 1. Ejemplos de error_reporting()

```
<?php

// Deshabilita la notificaci&oacute;n de errores
error_reporting(0);

// Notifica errores simples de ejecuci&oacute;n
error_reporting(E_ERROR | E_WARNING | E_PARSE);

// Utilizar E_NOTICE puede ser &uacute;til tambi&eacute;n (para notificar sobre
// variables no inicializadas y otros errores de escritura ...)
error_reporting(E_ERROR | E_WARNING | E_PARSE | E_NOTICE);

// Notifica todos los errores salvo E_NOTICE
// Este nivel es el valor por defecto establecido en el archivo php.ini
error_reporting(E_ALL ^ E_NOTICE);

// Notifica todos los errores de PHP (en PHP 3 se puede utilizar la m&aacute;scara 63)
error_reporting(E_ALL);

// Lo siguiente es ñalente a error_reporting(E_ALL);
ini_set('error_reporting', E_ALL);

?>
```

Los niveles de error disponibles se muestran a continuación. La explicación de cada uno de los niveles de error se encuentra en la sección .

Tabla 1. Constantes y máscaras de bits de los niveles de error_reporting()

valor	constante
1	E_ERROR
2	E_WARNING
4	E_PARSE
8	E_NOTICE

valor	constante
16	E_CORE_ERROR
32	E_CORE_WARNING
64	E_COMPILE_ERROR
128	E_COMPILE_WARNING
256	E_USER_ERROR
512	E_USER_WARNING
1024	E_USER_NOTICE
2047	E_ALL
2048	E_STRICT

Aviso
Las versiones de PHP > 5.0.0 también incluyen el nivel **`E_STRICT`** (valor 2048). **`E_ALL`** *NO* incluye el nivel **`E_STRICT`**.

Puede consultar también la directiva display_errors y la función **ini_set()**.

restore_error_handler

(PHP 4 >= 4.0.1, PHP 5)

restore_error_handler -- Recupera la anterior función de manejo de errores

Descripción

void **restore_error_handler** (void)

Después de cambiar la función que maneja los errores, se puede utilizar la función **set_error_handler()** para volver a la anterior función de manejo de errores (sea una función propia de PHP o una función definida por el usuario)

Puede consultar también las funciones **error_reporting()**, **set_error_handler()**, **trigger_error()** y **user_error()**

restore_exception_handler

(PHP 5)

restore_exception_handler -- Restores the previously defined exception handler function

Descripción

bool **restore_exception_handler** (void)

Used after changing the exception handler function using **set_exception_handler()**, to revert to the previous exception handler (which could be the built-in or a user defined function). This function always returns **`TRUE`**.

See also **set_exception_handler()**, **set_error_handler()**, **restore_error_handler()** **error_reporting()**

set_error_handler

(PHP 4 >= 4.0.1, PHP 5)

set_error_handler -- Establece una función específica para el manejo de errores.

Descripción

string **set_error_handler** (string error_handler)

Establece una función definida por el usuario (*error_handler*) para el manejo de los errores dentro de un script. Devuelve el anterior manejador de errores (si es que existe alguno) o **FALSE** si se produce un error. Esta función se utiliza para definir un método específico de manejo de los errores que se producen en tiempo de ejecución. Por ejemplo, se puede utilizar en aplicaciones que requieran un tratamiento especial de algunos datos/archivos cuando se produce un error crítico o cuando se lanza un determinado error (utilizando la función **trigger_error()**)

La función que se quiere utilizar para el manejo de errores debe aceptar 2 parámetros: el código de error y una cadena de texto que describe el error producido. A continuación se muestra un ejemplo de uso de excepciones mediante el lanzamiento de errores y su manejo con una función definida por el usuario:

Ejemplo 1. Manejo de errores con las funciones set_error_handler() y trigger_error()

```php
<?php

// Definición de las constantes de error del usuario - solo PHP4
define (FATAL,E_USER_ERROR);
define (ERROR,E_USER_WARNING);
define (WARNING,E_USER_NOTICE);

// Establecer el nivel de errores notificado en este script
error_reporting  (FATAL + ERROR + WARNING);

// Función de manejo de errores
function myErrorHandler ($errno, $errstr) {
    switch ($errno) {
    case FATAL:
    echo "<b>FATAL</b> [$errno] $errstr<br>\n";
    echo "  Fatal error in line ".__LINE__." of file ".__FILE__;
    echo ", PHP ".PHP_VERSION." (".PHP_OS.")<br>\n";
    echo "Aborting...<br>\n";
    exit -1;
    break;
    case ERROR:
    echo "<b>ERROR</b> [$errno] $errstr<br>\n";
    break;
    case WARNING:
    echo "<b>WARNING</b> [$errno] $errstr<br>\n";
    break;
    default:
    echo "Unkown error type: [$errno] $errstr<br>\n";
    break;
    }
}

// Función par probar el manejo de errores
function scale_by_log ($vect, $scale) {
    if ( !is_numeric($scale) || $scale <= 0 )
    trigger_error("log(x) for x <= 0 is undefined, you used: scale = $scale",
      FATAL);
    if (!is_array($vect)) {
    trigger_error("Incorrect input vector, array of values expected", ERROR);
    return null;
    }
    for ($i=0; $i<count($vect); $i++) {
    if (!is_numeric($vect[$i]))
    trigger_error("Value at position $i is not a number, using 0 (zero)",
      WARNING);
    $temp[$i] = log($scale) * $vect[$i];
    }
    return $temp;
}

// Establecer la función de manejo de errores
$old_error_handler = set_error_handler("myErrorHandler");

// Lanzamiento de algunos errores. En primer lugar se define un array mixto con un elem
echo "vector a\n";
$a = array(2,3,"foo",5.5,43.3,21.11);
print_r($a);

// Crear un nuevo array, generando un aviso
echo "----\nvector b - a warning (b = log(PI) * a)\n";
$b = scale_by_log($a, M_PI);
print_r($b);

// Lo siguiente produce un error, ya que se pasa una cadena de texto en lugar de un arr
echo "----\nvector c - an error\n";
$c = scale_by_log("not array",2.3);
var_dump($c);

// A continuación se produce un error crítico, ya que no está definido el logaritmo
// de un número negativo o igual a cero
echo "----\nvector d - fatal error\n";
$d = scale_by_log($a, -2.5);
```

Cuando se ejecuta el anterior ejemplo, el resultado será similar al siguiente:

```
vector a
Array
(
    [0] => 2
    [1] => 3
    [2] => foo
    [3] => 5.5
    [4] => 43.3
    [5] => 21.11
)
----
vector b - a warning (b = log(PI) * a)
<b>WARNING</b> [1024] Value at position 2 is not a number, using 0 (zero)<br />
Array
(
    [0] => 2.2894597716988
    [1] => 3.4341896575482
    [2] => 0
    [3] => 6.2960143721717
    [4] => 49.566804057279
    [5] => 24.165247890281
)
----
vector c - an error
<b>ERROR</b> [512] Incorrect input vector, array of values expected<br />
NULL
----
vector d - fatal error
<b>FATAL</b> [256] log(x) for x <= 0 is undefined, you used: scale = -2.5<br />
  Fatal error in line 36 of file trigger_error.php, PHP 4.0.2 (Linux)<br />
Aborting...<br />
```

Un aspecto importante a tener en cuenta cuando se utiliza un manejador propio es que el manejador de errores de PHP se deshabilita por completo (de forma temporal). De esta forma, los cambios efectuados con la función **error_reporting()** no tienen ningún efecto, ya que se utilizará exclusivamente la función manejadora de errores definida por el usuario. No obstante, se puede consultar el valor almacenado en error_reporting y actuar en consecuencia. Además, es necesario tener presente que el valor almacenado en error_reporting será igual a 0 si la sentencia que originó el error tiene como prefijo el operador @ .

Además, la función manejadora de errores deberá hacer uso de la función **die()** si es necesario. Si la función manejadora de errores deuelve un valor, la ejecuciÃƒÆ'Ã‚Â³n del script continuará en la sentencia siguiente a la que originó el error.

> **Nota:** Si se produce un error antes de que se ejecute el script (por ejemplo cuando se sube un archivo), no se puede utilizar un manejador creado por el usuario, ya que en ese momento aún no ha sido registrado.

> **Nota:** El segundo parámetro *error_types* se introdujo a partir de PHP 5.

Puede consultar tambié las funciones **error_reporting()**, **restore_error_handler()**, **trigger_error()** y **user_error()**

set_exception_handler

(PHP 5)

set_exception_handler -- Sets a user-defined exception handler function

Descripción

string **set_exception_handler** (callback exception_handler)

Sets the default exception handler if an exception is not caught within a try/catch block. Execution will stop after the *exception_handler* is called.

The *exception_handler* must be defined before calling **set_exception_handler()**. This function needs to accept one parameter, which will be the exception object that was thrown.
exception_handler (object exception)

exception

Name of function to be called when an uncaught exception occurs.

Lista de parámetros

exception_handler

Name of function to be called when an uncaught exception occurs.

Valores retornados

Returns the previously defined exception handler, or **FALSE** on error. If no previous handler was defined, an empty string is returned.

Ejemplos

Ejemplo 1. set_exception_handler() example

```
<?php
function exception_handler($exception) {
  echo "Uncaught exception: " , $exception->getMessage(), "\n";
}

set_exception_handler('exception_handler');

throw new Exception('Uncaught Exception');
echo "Not Executed\n";
?>
```

Ver también

restore_exception_handler(), **restore_error_handler()**, **error_reporting()**, information about the callback type, , y PHP 5 Exceptions.

trigger_error

(PHP 4 >= 4.0.1, PHP 5)

trigger_error -- Genera un mensaje de error/aviso/notificación a nivel de usuario

Descripción

bool **trigger_error** (string error_msg [, int error_type])

Esta función se utiliza generalmente para lanzar mensajes de error. Se puede usar junto con el manejador de errores interno de PHP o con funciones manejadoras de error creadas por el usuario (**set_error_handler()**). Solamente funciona con la familia de constantes E_USER y por defecto lo hará con **E_USER_NOTICE**.

El valor devuelto será **FALSE** si se especifica un valor de *error_type* incorrecto y **TRUE** en cualquier otro caso.

La mayor utilidad de esta función es la de generar respuestas personalizadas a las excepciones en tiempo de ejecución. Por ejemplo:

```
<?php
if (assert($divisor == 0)) {
  trigger_error("Cannot divide by zero", E_USER_ERROR);
}
?>
```

> **Nota:** Puede consultar la función **set_error_handler()** para ver un ejemplo mÃ‚Â´s extenso.
>
> **Nota:** El parámetro *error_msg* tiene un límite de 1024 caracteres. La función trunca cualquier valor superior a 1024.

Puede consultar también las funciones **error_reporting()**, **set_error_handler()**, **restore_error_handler()** y las constantes de nivel de error.

user_error

user_error -- Alias de **trigger_error()**

Descripción

Se trata de un alias de la función **trigger_error()**.

XXXI. Funciones de Ejecución de Programas

Introducción

Estas funciones proveen medios para ejecutar comandos en el sistema mismo, y medios para proveer seguridad con tales comandos.

Requirimientos

No se necesitan bibliotecas externas para construir esta extensión

---

Instalación

No se necesita ninguna instalación para usar estas funciones, son parte del núcleo de PHP.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Esta extensión no tiene ninguna constante definida.

Ver también

Estas funciones se relacionan estrechamente también con el operador de comilla invertida.
Asimismo, cuando se encuentre bajo safe mode, debe considerar la directiva safe_mode_exec_dir.

Tabla de contenidos

escapeshellarg

(PHP 4 >= 4.0.3, PHP 5)

escapeshellarg -- Escapar una cadena a ser usada como argumento del intérprete de comandos

Descripción

string **escapeshellarg** (string arg)

escapeshellarg() agrega comillas sencillas alrededor de una cadena y rodea de comillas/escapa cualquier comilla sencilla existente, permitiéndole pasar la cadena directamente a una función del intérprete de comandos, y logrando que sea tratada como un solo argumento seguro. Esta función debe ser usada para escapar argumentos individuales a funciones del intérprete de comandos que provienen de la entrada del usuario. Las funciones del intérprete de comandos incluyen **exec()**, **system()** y el operador de comilla invertida. Una forma regular de usar esta función sería:

```
<?php
system('ls '.escapeshellarg($dir));
?>
```

Vea también **escapeshellcmd()**, **exec()**, **popen()**, **system()**, y el operador de comillas invertidas.

escapeshellcmd

(PHP 3, PHP 4 , PHP 5)

escapeshellcmd -- enmascara los metacaracteres del intérprete de ordenes

Descripción

string **escapeshellcmd** (string command)

EscapeShellCmd() enmascara cualquier carácter en una cadena de caracteres que pueda usarse para introducir fraudulentamente una orden al intérprete de órdenes para que éste ejecute instrucciones arbitrarias. Esta función se debería usar para asegurarse que cualquier dato que venga del usuario se enmascare antes de que éste se le pase a las funciones **exec()** o **system()**, o al operador ` (apóstrofe invertido) . Un uso habitual podría ser:

```
system(EscapeShellCmd($cmd))
```

Véase también **exec()**, **popen()**, **system()**, y el operador ` (apóstrofe invertido).

exec

(PHP 3, PHP 4 , PHP 5)

exec -- Ejecuta un programa externo

Descripción

string **exec** (string command [, string array [, int return_var]])

exec() ejecuta la orden indicada en *command*, sin embargo no produce ninguna salida. Simplemente devuelve la última línea de la salida resultado de la orden. Si necesita ejecutar una orden y obtener directamente todos los datos devueltos por la orden sin ninguna interferencia, use la función **PassThru()**.

Si el parámetro *array* existe, entonces el array especificado se rellenará con cada una de las líneas de la salida producida por la orden. Notar que si el array ya contiene algunos elementos, **exec()** los añadirá al final del array. Si no quiere que la función añada dichos elementos, haga un **unset()** sobre el array antes de pasárselo a **exec()**.

Si el parámetro *return_var* existe a la vez que el parámetro *array*, entonces el valor de retorno de la orden ejecutada se guardará en dicha variable.

Destacar que si usted va a permitir que se pasen datos provenientes de usuarios a esta función, entonces debería usar **EscapeShellCmd()** para asegurarse de que los usuarios no pueden engañar al sistema para ejecutar instrucciones arbitrarias.

Véase también **system()**, **PassThru()**, **popen()**, **EscapeShellCmd()**, y el operador ` (apóstrofe invertido).

passthru

(PHP 3, PHP 4 , PHP 5)

passthru -- Ejecuta un programa externo y muestra su salida literal

Descripción

string **passthru** (string command [, int return_var])

La función **passthru()** es similar a la función **exec()** en que ejecuta una orden (*command*). Si existe el parámetro *return_var*, el valor de estado devuelto por la orden Unix se guardará ahí. Esta función debería usarse en lugar de **exec()** o **system()** cuando la salida de la orden Unix sean datos binarios que deban ser pasados directamente al navegador. Un uso típico de ello es ejecutar algo como las utilidades pbmplus las cuales pueden dar como resultado directamente el flujo de datos de una imagen. Poniendo el content-type a *image/gif* y llamando al programa pbmplus para mostrar un gif, usted puede crear archivos de órdenes PHP que generen directamente imágenes.

Véase también **exec()**, **system()**, **popen()**, **EscapeShellCmd()**, y el operador ` (apóstrofe invertido).

proc_close

(PHP 4 >= 4.3.0, PHP 5)

proc_close -- Cierra un proceso abierto por **proc_open()** y devuelve el código de salida del proceso.

Descripción

int **proc_close** (resource proceso)

proc_close() es similar a **pclose()**, excepto que solo trabaja con procesos abiertos por **proc_open()**. **proc_close()** espera a que el proceso termine, y devuelve su código de salida. Si tiene pipes abiertos con ese proceso, debe usar **fclose()** sobre ellos antes de llamar esta función, con el propósito de evitar bloqueos muertos - puede que el proceso hijo no pueda salir mientras los pipes se encuentren abiertos.

proc_get_status

(PHP 5)

proc_get_status -- Obtiene información sobre un proceso abierto por **proc_open()**

Descripción

array **proc_get_status** (resource proceso)

proc_get_status() recupera información sobre un proceso abierto mediante el uso de **proc_open()**. Los datos recolectados son devueltos en una matriz que contiene los siguientes elementos:

elemento	tipo	descripción
command	**string**	la cadena del comando que fue pasada a **proc_open()**
pid	**int**	id del proceso
running	**bool**	**TRUE** si el proceso aun está siendo ejecutado, **FALSE** si ha terminado
signaled	**bool**	**TRUE** si el proceso hijo ha sido terminado por una señal no atrapada. En windows, este valor siempre es **FALSE**.
stopped	**bool**	**TRUE** si el proceso hijo fue detenido por una señal. En windows, este valor siempre es **FALSE**.
exitcode	**int**	el código de salida devuelto por el proceso (el cual tiene sentido únicamente si *running* es **FALSE**)
termsig	**int**	el número de la señal que causó que el proceso hijo finalizara su ejecución (sólo tiene sentido si *signaled* es **TRUE**)
stopsig	**int**	el número de la señal que causó que el proceso hijo detuviera su ejecución (sólo tiene sentido si *stopped* es **TRUE**)

Vea también **proc_open()**.

proc_nice

(PHP 5)

proc_nice -- Modificar la prioridad del proceso actual

Descripción

bool **proc_nice** (int incremento)

proc_nice() modifica la prioridad del proceso actual por la cantidad especificada en *incremento*. Un *incremento* positivo reducirá la prioridad del proceso actual, mientras que un *incremento* negativo la incrementará. Si ocurre un error, como que el usuario carezca de permisos para modificar la prioridad, un error de nivel **E_WARNING** es generado, y se devuelve **FALSE**. De otro modo, se devuelve **TRUE**.

Nota: proc_nice() existirá únicamente si su sistema tiene soporte para 'nice'. 'nice' está

definido de acuerdo a los estándares: SVr4, SVID EXT, AT&T, X/OPEN, BSD 4.3. Esto quiere decir que **proc_nice()** no está disponible en windows.

La función **proc_nice()** no se encuentra relacionada con **proc_open()** ni sus funciones asociadas en ninguna forma.

proc_open

(PHP 4 >= 4.3.0, PHP 5)

proc_open -- Ejecutar un comando y abrir apuntadores de archivo para entrada/salida

Descripción

resource **proc_open** (string cmd, array espec_descriptor, array &pipes [, string cwd [, array env [, array otras_opciones]]])

proc_open() es similar a **popen()** pero provee un grado de control mucho mayor sobre la ejecución del programa. *cmd* es el comando a ser ejecutado por el intérprete de comandos. *espec_descriptor* es una matriz indexada en donde la clave representa el número de descriptor y el valor representa el modo como PHP pasará ese descriptor al proceso hijo. *pipes* será definido como una matriz indexada de apuntadores a archivo que corresponden a los puntos de comunicación con PHP de todo pipe que sea creado. El valor de retorno es un recurso que representa el proceso; usted debe liberarlo usando **proc_close()** una vez haya terminado de usarlo.

```
<?php
$espec_descriptor = array(
   0 => array("pipe", "r"),  // stdin es un pipe usado por el hijo para lectura
   1 => array("pipe", "w"),  // stdout es un pipe usado por el hijo para escritura
   2 => array("file", "/tmp/error-output.txt", "a") // stderr es un archivo para escrit
);
$proceso = proc_open("php", $espec_descriptor, $pipes);
if (is_resource($proceso)) {
    // $pipes ahora luce de esta forma:
    // 0 => gestor de escritura conectado con la entrada estandar del hijo
    // 1 => gestor de lectura conectado con la salida estandar del hijo
    // Cualquier mensaje de salida de error sera adicionado a /tmp/error-output.txt

    fwrite($pipes[0], "<?php echo \"&iexcl;Hola mundo!\"; ?>");
    fclose($pipes[0]);

    while (!feof($pipes[1])) {
        echo fgets($pipes[1], 1024);
    }
    fclose($pipes[1]);
    // Es importante que cierre todos los pipes antes de llamar
    // proc_close para evitar un bloqueo muerto
    $retval = proc_close($proceso);

    echo "el comando ha devuelto $retval\n";
}
?>
```

PHP 5RC2 introduce soporte pty para sistemas con ptys Unix98. Esto le permite a su script interactuar con aplicaciones que esperan estar hablando con una terminal. Una pty trabaja como un pipe, pero es bi-direccional, así que no hay necesidad de especificar un modo de lectura/escritura. El siguiente ejemplo muestra cómo usar una pty; note que no necesita tener todos los descriptores hablando con una pty. Note también que solo una pty es creada, incluso cuando pty se especifica 3 veces. En una versión futura de PHP, puede que sea posible hacer más que simplemente leer y

escribir a la pty.

```
<?php
// Crear una pseudo terminal para el proceso hijo
$espec_descriptor = array(
   0 => array("pty"),
   1 => array("pty"),
   2 => array("pty")
);
$proceso = proc_open("cvs -d:pserver:cvsread@cvs.php.net:/repository login", $espec_des
if (is_resource($proceso)) {
   // trabaje con el recurso aqui
}
?>
```

Los números de descriptor de archivo en *espec_descriptor* no están limitados a 0, 1 y 2 - usted puede especificar cualquier número de descriptor de archivo válido y éste será pasado al proceso hijo. Esto le permite a su script interoperar con otros scripts que corran como "co-procesos". En particular, esto es útil para pasar contraseñas a programas como PGP, GPG y openssl en un modo más seguro. También es útil para la lectura de información de status entregada por aquellos programas en descriptores de archivo auxiliares.

> **Nota:** Compatibilidad con windows: Los descriptores más allá de 2 (stderr) son entregados al proceso hijo como gestores heredables, pero ya que la arquitectura windows no asocia números de descriptor de archivo con gestores de bajo nivel, el proceso hijo no dispone (aun) de un medio para acceder a esos gestores. Stdin, stdout y stderr funcionan como es de esperar.

> **Nota:** Si sólo necesita un pipe de proceso uni-direccional (una-vía), use **popen()** en su lugar, ya que es mucho más fácil de usar.

Vea también **stream_select()**, **exec()**, **system()**, **passthru()**, **popen()**, **escapeshellcmd()**, y el operador de comilla invertida.

proc_terminate

(PHP 5)

proc_terminate -- mata un proceso abierto por proc_open

Descripción

int **proc_terminate** (resource proceso [, int senyal])

Envía una señal al *proceso* (creado usando **proc_open()**) indicando que debería terminar. **proc_terminate()** retorna inmediatamente y no espera a que el proceso termine.

La *senyal* opcional es útil únicamente en sistemas operativos POSIX; puede especificar una señal a ser enviada al proceso usando la llamada del sistema *kill(2)*. La señal predeterminada es *SIGTERM*.

proc_terminate() le permite terminar el proceso y continuar con otras tareas. Puede consultar el estado del proceso (para ver si ya se ha detenido) usando la función **proc_get_status()**.

Vea también **proc_open()**, **proc_close()**, y **proc_get_status()**.

shell_exec

(PHP 4 , PHP 5)

shell_exec -- Ejecutar un comando mediante el intérprete de comandos y devolver la salida completa como una cadena

Descripción

string **shell_exec** (string cmd)

Esta función es idéntica al operador de comillas invertidas.

Ejemplo 1. Un ejemplo de shell_exec()

```
<?php
$salida = shell_exec('ls -lart');
echo "<pre>$salida</pre>";
?>
```

Nota: Esta función no está habilitada en safe-mode (modo-seguro)

Vea también **exec()** y **escapeshellcmd()**.

system

(PHP 3, PHP 4 , PHP 5)

system -- Ejecutar un programa externo y mostrar su salida

Descripción

string **system** (string comando [, int &val_retorno])

system() es similar a la versión C de la función de mismo nombre, dado que ejecuta el *comando* dado y muestra el resultado. Si se entrega una variable como segundo argumento, entonces el código de status devuelto por el comando ejecutado será escrito en esta variable.

Aviso
Si se va a permitir que datos provenientes del usuario sean enviados a esta funcion, habria que utilizar **escapeshellarg()** o **escapeshellcmd()** para asegurarse que el usuario no intenta engañar al sistema para que ejecute comandos arbitrarios.

Nota: Si arrancamos un programa con esta funcion y queremos dejarlo ejecutandose en segundo plano, hay que asegurarse que el resultado del mismo es redireccionado a un fichero u otra salida o PHP se parara hasta que la ejecucion del programa termine.

La llamada a **system()** también intenta volcar automáticamente el búfer de salida del servidor web después de cada línea de salida, si PHP está corriendo como un módulo de servidor.

Devuelve la última línea de la salida del comando en caso de éxito, y **FALSE** si se presenta algún fallo.

Si necesita ejecutar un comando y recibir de vuelta todo los datos del mismo sin interferencias, use la función **passthru()**.

Ejemplo 1. Ejemplo de system()

```
<?php
echo '<pre>';

// Muestra el resultado completo del comando "ls", y devuelve la
// ultima linea de la salida en $ultima_linea. Almacena el valor de
// retorno del comando en $retval.
$ultima_linea = system('ls', $retval);

// Imprimir informacion adicional
echo '
</pre>
<hr />Ultima linea de la salida: ' . $ultima_linea . '
<hr />Valor de retorno: ' . $retval;
?>
```

> **Nota:** Cuando safe mode esta activado, solamente se pueden ejecutar los programas que se encuentren en safe_mode_exec_dir. Por razones practicas, no se permite el uso de .. en el PATH del programa.

Aviso
Con safe mode activado, todas las palabras que siguan al comando inicial son tratadas como un solo argumento. Asi, *echo y | echo x* se interpreta como *echo "y | echo x"*.

Vea también **exec()**, **passthru()**, **popen()**, **escapeshellcmd()**, **pcntl_exec()** y el operador de comilla invertida.

XXXII. Exif Functions

Introducción

With the exif extension you are able to work with image meta data. For example, you may use exif functions to read meta data of pictures taken from digital cameras by working with information stored in the headers of the JPEG and TIFF images.

Requirimientos

Your PHP must be compiled in with *--enable-exif*. PHP does not require any additional library for the exif module. Windows users must also have the mbstring extension enabled.

Instalación

To enable exif-support configure PHP with *--enable-exif*

Windows users must enable both the `php_mbstring.dll` and `php_exif.dll` DLL's in `php.ini`. The `php_mbstring.dll` DLL must be loaded *before* the `php_exif.dll` DLL so adjust your `php.ini` accordingly.

Configuración en tiempo de ejecución

El comportamiento de estas funciones está afectado por los valores definidos en `php.ini`.

Exif supports automatically conversion for Unicode and JIS character encodings of user comments when module mbstring is available. This is done by first decoding the comment using the specified characterset. The result is then encoded with another characterset which should match your *HTTP* output.

Tabla 1. Exif configuration options

Name	Default	Changeable
exif.encode_unicode	"ISO-8859-15"	PHP_INI_ALL
exif.decode_unicode_motorola	"UCS-2BE"	PHP_INI_ALL
exif.decode_unicode_intel	"UCS-2LE"	PHP_INI_ALL
exif.encode_jis	""	PHP_INI_ALL
exif.decode_jis_motorola	"JIS"	PHP_INI_ALL
exif.decode_jis_intel	"JIS"	PHP_INI_ALL

For further details and definitions of the PHP_INI_* constants, see the Apéndice H.

A continuación se presenta una corta explicación de las directivas de configuración.

exif.encode_unicode **string**

> *exif.encode_unicode* defines the characterset UNICODE user comments are handled. This defaults to ISO-8859-15 which should work for most non Asian countries. The setting can be empty or must be an encoding supported by mbstring. If it is empty the current internal encoding of mbstring is used.

exif.decode_unicode_motorola **string**

> *exif.decode_unicode_motorola* defines the image internal characterset for Unicode encoded user comments if image is in motorola byte order (big-endian). This setting cannot be empty but you can specify a list of encodings supported by mbstring. The default is UCS-2BE.

exif.decode_unicode_intel **string**

> *exif.decode_unicode_intel* defines the image internal characterset for Unicode encoded user comments if image is in intel byte order (little-endian). This setting cannot be empty but you can specify a list of encodings supported by mbstring. The default is UCS-2LE.

exif.encode_jis **string**

> *exif.encode_jis* defines the characterset JIS user comments are handled. This defaults to an empty value which forces the functions to use the current internal encoding of mbstring.

exif.decode_jis_motorola **string**

> *exif.decode_jis_motorola* defines the image internal characterset for JIS encoded user

comments if image is in motorola byte order (big-endian). This setting cannot be empty but you can specify a list of encodings supported by mbstring. The default is JIS.

exif.decode_jis_intel **string**

exif.decode_jis_intel defines the image internal characterset for JIS encoded user comments if image is in intel byte order (little-endian). This setting cannot be empty but you can specify a list of encodings supported by mbstring. The default is JIS.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

`EXIF_USE_MBSTRING` (**integer**)

The **exif_imagetype()** lists several related built-in constants.

Tabla de contenidos

exif_imagetype

(PHP 4 >= 4.3.0, PHP 5)

exif_imagetype -- Determine the type of an image

Descripción

int **exif_imagetype** (string filename)

exif_imagetype() reads the first bytes of an image and checks its signature.

exif_imagetype() can be used to avoid calls to other exif functions with unsupported file types or in conjunction with *$_SERVER['HTTP_ACCEPT']* to check whether or not the viewer is able to see a specific image in the browser.

Lista de parámetros

filename

The image being checked.

Valores retornados

When a correct signature is found, the appropriate constant value will be returned otherwise the return value is **FALSE**. The return value is the same value that **getimagesize()** returns in index 2 but **exif_imagetype()** is much faster.

Registro de cambios

Versión	Descripción
4.3.2	Support for JPC, JP2, JPX, JB2, XBM, and WBMP
4.3.0	Support for SWC

Constantes predefinidas

The following constants are defined, and represent possible **exif_imagetype()** return values:

Tabla 1. Imagetype Constants

Value	Constant
1	**IMAGETYPE_GIF**
2	**IMAGETYPE_JPEG**
3	**IMAGETYPE_PNG**
4	**IMAGETYPE_SWF**
5	**IMAGETYPE_PSD**
6	**IMAGETYPE_BMP**
7	**IMAGETYPE_TIFF_II** (intel byte order)
8	**IMAGETYPE_TIFF_MM** (motorola byte order)
9	**IMAGETYPE_JPC**
10	**IMAGETYPE_JP2**
11	**IMAGETYPE_JPX**
12	**IMAGETYPE_JB2**
13	**IMAGETYPE_SWC**
14	**IMAGETYPE_IFF**
15	**IMAGETYPE_WBMP**
16	**IMAGETYPE_XBM**

Ejemplos

Ejemplo 1. exif_imagetype() example

```
<?php
if (exif_imagetype('image.gif') != IMAGETYPE_GIF) {
    echo 'The picture is not a gif';
}
?>
```

Ver también

getimagesize()

exif_read_data

(PHP 4 >= 4.2.0, PHP 5)

exif_read_data -- Reads the EXIF headers from JPEG or TIFF

Descripción

array **exif_read_data** (string filename [, string sections [, bool arrays [, bool thumbnail]]])

exif_read_data() reads the EXIF headers from a JPEG or TIFF image file. This way you can read meta data generated by digital cameras.

Exif headers tend to be present in JPEG/TIFF images generated by digital cameras, but unfortunately each digital camera maker has a different idea of how to actually tag their images, so you can't always rely on a specific Exif header being present.

Height and *Width* are computed the same way **getimagesize()** does so their values must not be part of any header returned. Also, *html* is a height/width text string to be used inside normal HTML.

When an Exif header contains a Copyright note, this itself can contain two values. As the solution is inconsistent in the Exif 2.10 standard, the COMPUTED section will return both entries *Copyright.Photographer* and *Copyright.Editor* while the IFD0 sections contains the byte array with the NULL character that splits both entries. Or just the first entry if the datatype was wrong (normal behaviour of Exif). The COMPUTED will also contain the entry *Copyright* which is either the original copyright string, or a comma separated list of the photo and editor copyright.

The tag UserComment has the same problem as the Copyright tag. It can store two values. First the encoding used, and second the value itself. If so the IFD section only contains the encoding or a byte array. The COMPUTED section will store both in the entries *UserCommentEncoding* and *UserComment*. The entry *UserComment* is available in both cases so it should be used in preference to the value in IFD0 section.

> **Nota:** Windows ME/XP can both wipe the Exif headers when connecting to a camera. More information available at http://www.canon.co.jp/Imaging/NOTICE/011214-e.html.

Lista de parámetros

filename

The name of the image file being read. This cannot be an URL.

sections

Is a comma separated list of sections that need to be present in file to produce a result **array**. If none of the requested sections could be found the return value is **FALSE**.

FILE	FileName, FileSize, FileDateTime, SectionsFound
COMPUTED	html, Width, Height, IsColor, and more if available. Height and Width are computed the same way **getimagesize()** does so their values must not be part of any header returned. Also, html is a height/width text string to be used inside normal HTML.
ANY_TAG	Any information that has a Tag e.g. IFD0, EXIF, ...
IFD0	All tagged data of IFD0. In normal imagefiles this contains image size and so forth.
THUMBNAIL	A file is supposed to contain a thumbnail if it has a second IFD. All tagged information about the embedded thumbnail is stored in this section.
COMMENT	Comment headers of JPEG images.
EXIF	The EXIF section is a sub section of IFD0. It contains more detailed information about an image. Most of these entries are digital camera related.

arrays

Specifies whether or not each section becomes an array. The *sections* *COMPUTED*, *THUMBNAIL*, and *COMMENT* always become arrays as they may contain values whose names conflict with other sections.

thumbnail

When set to **TRUE** the thumbnail itself is read. Otherwise, only the tagged data is read.

Valores retornados

It returns an associative **array** where the array indexes are the header names and the array values are the values associated with those headers. If no data can be returned, **exif_read_data()** will return **FALSE**.

Registro de cambios

Versión	Descripción
4.3.0	Can read all embedded IFD data including arrays (returned as such). Also the size of an embedded thumbnail is returned in a *THUMBNAIL* subarray, and can return thumbnails in TIFF format. Also, there is no longer a maximum length for returned values (not until the memory limit has been reached)

Versión	Descripción
4.3.0	If PHP has mbstring support, the user comment can automatically change encoding. Also, if the user comment uses Unicode or JIS encoding this encoding will automatically be changed according to the exif ini settings in `php.ini`
4.3.0	If the image contains any IFD0 data then COMPUTED contains the entry ByteOrderMotorola which is 0 for little-endian (intel) and 1 for big-endian (motorola) byte order. Also, COMPUTED and UserComment no longer only contain the first copyright entry if the datatype was wrong.

Ejemplos

Ejemplo 1. exif_read_data() example

```
<?php
echo "test1.jpg:<br />\n";
$exif = exif_read_data('tests/test1.jpg', 'IFD0');
echo $exif===false ? "No header data found.<br />\n" : "Image contains headers<br />\n"

$exif = exif_read_data('tests/test2.jpg', 0, true);
echo "test2.jpg:<br />\n";
foreach ($exif as $key => $section) {
    foreach ($section as $name => $val) {
        echo "$key.$name: $val<br />\n";
    }
}
?>
```

The first call fails because the image has no header information.

El resultado del ejemplo seria algo similar a:

```
test1.jpg:
No header data found.
test2.jpg:
FILE.FileName: test2.jpg
FILE.FileDateTime: 1017666176
FILE.FileSize: 1240
FILE.FileType: 2
FILE.SectionsFound: ANY_TAG, IFD0, THUMBNAIL, COMMENT
COMPUTED.html: width="1" height="1"
COMPUTED.Height: 1
COMPUTED.Width: 1
COMPUTED.IsColor: 1
COMPUTED.ByteOrderMotorola: 1
COMPUTED.UserComment: Exif test image.
COMPUTED.UserCommentEncoding: ASCII
COMPUTED.Copyright: Photo (c) M.Boerger, Edited by M.Boerger.
COMPUTED.Copyright.Photographer: Photo (c) M.Boerger
COMPUTED.Copyright.Editor: Edited by M.Boerger.
IFD0.Copyright: Photo (c) M.Boerger
IFD0.UserComment: ASCII
THUMBNAIL.JPEGInterchangeFormat: 134
THUMBNAIL.JPEGInterchangeFormatLength: 523
COMMENT.0: Comment #1.
COMMENT.1: Comment #2.
COMMENT.2: Comment #3end
THUMBNAIL.JPEGInterchangeFormat: 134
THUMBNAIL.Thumbnail.Height: 1
THUMBNAIL.Thumbnail.Height: 1
```

Ver también

exif_thumbnail()

getimagesize()

exif_tagname

(PHP 4 >= 4.2.0, PHP 5)

exif_tagname -- Get the header name for an index

Descripción

string **exif_tagname** (string index)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

Lista de parámetros

index

The image index

Valores retornados

Returns the header name, or **FALSE** if *index* is undefined.

Ver también

exif_imagetype()

exif_thumbnail

(PHP 4 >= 4.2.0, PHP 5)

exif_thumbnail -- Retrieve the embedded thumbnail of a TIFF or JPEG image

Descripción

string **exif_thumbnail** (string filename [, int &width [, int &height [, int &imagetype]]])

exif_thumbnail() reads the embedded thumbnail of a TIFF or JPEG image.

If you want to deliver thumbnails through this function, you should send the mimetype information using the **header()** function.

It is possible that **exif_thumbnail()** cannot create an image but can determine its size. In this case, the return value is **FALSE** but *width* and *height* are set.

Lista de parámetros

filename

The name of the image file being read. This image contains an embedded thumbnail.

width

The return width of the returned thumbnail.

height

The returned height of the returned thumbnail.

imagetype

The returned image type of the returned thumbnail. This is either TIFF or JPEG.

Valores retornados

Returns the embedded thumbnail, or **`FALSE`** if the image contains no thumbnail.

Registro de cambios

Versión	Descripción
4.3.0	The optional parameters *width*, *height*, and *imagetype* all became available.
4.3.0	May return thumbnails in the TIFF format.

Ejemplos

Ejemplo 1. exif_thumbnail() example

```
<?php
if (array_key_exists('file', $_REQUEST)) {
    $image = exif_thumbnail($_REQUEST['file'], $width, $height, $type);
} else {
    $image = false;
}
if ($image!==false) {
    header('Content-type: ' .image_type_to_mime_type($type));
    echo $image;
    exit;
} else {
    // no thumbnail available, handle the error here
    echo 'No thumbnail available';
}
?>
```

Ver también

exif_read_data()
image_type_to_mime_type()

read_exif_data

read_exif_data -- Alias of **exif_read_data()**

Descripción

This function is an alias of **exif_read_data()**.

XXXIII. File Alteration Monitor Functions

Introducción

FAM monitors files and directories, notifying interested applications of changes. More information about FAM is available at http://oss.sgi.com/projects/fam/.

A PHP script may specify a list of files for FAM to monitor using the functions provided by this extension.

The FAM process is started when the first connection from any application to it is opened. It exits after all connections to it have been closed.

> **Nota:** Esta extensión no está disponible en plataformas Windows

---

Requirimientos

This extension uses the functions of the FAM library, developed by SGI. Therefore you have to download and install the FAM library.

Instalación

To use PHP's FAM support you must compile PHP *--with-fam[=DIR]* where DIR is the location of the directory containing the lib and include directories.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

FAM resource

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

Tabla 1. FAM event constants

Constant	Description
FAMChanged (integer)	Some value which can be obtained with fstat(1) changed for a file or directory.
FAMDeleted (integer)	A file or directory was deleted or renamed.
FAMStartExecuting (integer)	An executable file started executing.
FAMStopExecuting (integer)	An executable file that was running finished.
FAMCreated (integer)	A file was created in a directory.
FAMMoved (integer)	This event never occurs.
FAMAcknowledge (integer)	An event in response to fam_cancel_monitor().
FAMExists (integer)	An event upon request to monitor a file or directory. When a directory is monitored, an event for that directory and every file contained in that directory is issued.
FAMEndExist (integer)	Event after the last FAMEExists event.

Tabla de contenidos

fam_cancel_monitor

(PHP 5)

fam_cancel_monitor -- Terminate monitoring

Descripción

bool **fam_cancel_monitor** (resource fam, resource fam_monitor)

fam_cancel_monitor() terminates monitoring on a resource previously requested using one of the *fam_monitor_* functions. In addition an **FAMAcknowledge** event occurs.

See also **fam_monitor_file()**, **fam_monitor_directory()**, **fam_monitor_collection()**, and **fam_suspend_monitor()**

fam_close

(PHP 5)

fam_close -- Close FAM connection

Descripción

void **fam_close** (resource fam)

fam_close() closes a connection to the FAM service previously opened using **fam_open()**.

fam_monitor_collection

(PHP 5)

fam_monitor_collection -- Monitor a collection of files in a directory for changes

Descripción

resource **fam_monitor_collection** (resource fam, string dirname, int depth, string mask)

fam_monitor_collection() requests monitoring for a collection of files within a directory. The actual files to be monitored are specified by a directory path in *dirname*, the maximum search *depth* starting from this directory and a shell pattern *mask* restricting the file names to look for.

A FAM event will be generated whenever the status of the files change. The possible event codes are described in detail in the constants part of this section.

See also **fam_monitor_file()**, **fam_monitor_directory()**, **fam_cancel_monitor()**, **fam_suspend_monitor()**, and **fam_resume_monitor()**.

fam_monitor_directory

(PHP 5)

fam_monitor_directory -- Monitor a directory for changes

Descripción

resource **fam_monitor_directory** (resource fam, string dirname)

fam_monitor_directory() requests monitoring for a directory and all contained files. A FAM event will be generated whenever the status of the directory (i.e. the result of function **stat()** on that directory) or its content (i.e. the results of **readdir()**) change.

The possible event codes are described in detail in the constants part of this section.

See also **fam_monitor_file()**, **fam_monitor_collection()**, **fam_cancel_monitor()**, **fam_suspend_monitor()**, and **fam_resume_monitor()**.

fam_monitor_file

(PHP 5)

fam_monitor_file -- Monitor a regular file for changes

Descripción

resource **fam_monitor_file** (resource fam, string filename)

fam_monitor_file() requests monitoring for a single file. A FAM event will be generated whenever the file status (i.e. the result of function **stat()** on that file) changes.

The possible event codes are described in detail in the constants part of this section.

See also **fam_monitor_directory()**, **fam_monitor_collection()**, **fam_cancel_monitor()**, **fam_suspend_monitor()**, and **fam_resume_monitor()**.

fam_next_event

(PHP 5)

fam_next_event -- Get next pending FAM event

Descripción

array **fam_next_event** (resource fam)

fam_next_event() returns the next pending FAM event. The function will block until an event is available which can be checked for using **fam_pending()**.

fam_next_event() will return an array that contains a FAM event code in element '*code*', the path of the file this event applies to in element '*filename*' and optionally a hostname in element '*hostname*'.

The possible event codes are described in detail in the constants part of this section.

See also **fam_pending()**.

fam_open

(PHP 5)

fam_open -- Open connection to FAM daemon

Descripción

resource **fam_open** ([string appname])

fam_open() opens a connection to the FAM service daemon. The optional parameter *appname* should be set to a string identifying the application for logging reasons.

See also **fam_close()**.

fam_pending

(PHP 5)

fam_pending -- Check for pending FAM events

Descripción

bool **fam_pending** (resource fam)

fam_pending() returns **TRUE** if events are available to be fetched using **fam_next_event()**.

See also **fam_next_event()**.

fam_resume_monitor

(PHP 5)

fam_resume_monitor -- Resume suspended monitoring

Descripción

bool **fam_resume_monitor** (resource fam, resource fam_monitor)

fam_resume_monitor() resumes monitoring of a resource previously suspend using **fam_suspend_monitor()**.

See also **fam_suspend_monitor()**.

fam_suspend_monitor

(PHP 5)

fam_suspend_monitor -- Temporarily suspend monitoring

Descripción

bool **fam_suspend_monitor** (resource fam, resource fam_monitor)

fam_suspend_monitor() temporarily suspend monitoring of a resource previously requested using

one of the *fam_monitor_* functions. Monitoring can later be continued using **fam_resume_monitor()** without the need of requesting a complete new monitor.

See also **fam_resume_monitor()**, and **fam_cancel_monitor()**.

XXXIV. FrontBase Functions

Introducción

These functions allow you to access FrontBase database servers. More information about FrontBase can be found at http://www.frontbase.com/.

Documentation for FrontBase can be found at http://www.frontbase.com/cgi-bin/WebObjects/FrontBase.woa/wa/productsPage?currentPage=Documentation.

Frontbase support has been added to PHP 4.0.6.

Requirimientos

You must install the FrontBase database server or at least the fbsql client libraries to use this functions. You can get FrontBase from http://www.frontbase.com/.

Instalación

In order to have these functions available, you must compile PHP with fbsql support by using the *--with-fbsql[=DIR]* option. If you use this option without specifying the path to fbsql, PHP will search for the fbsql client libraries in the default installation location for the platform. Users who installed FrontBase in a non standard directory should always specify the path to fbsql: *--with-fbsql=/path/to/fbsql*. This will force PHP to use the client libraries installed by FrontBase, avoiding any conflicts.

Configuración en tiempo de ejecución

El comportamiento de estas funciones está afectado por los valores definidos en `php.ini`.

Tabla 1. FrontBase configuration options

Name	Default	Changeable
fbsql.allow_persistent	"1"	PHP_INI_SYSTEM
fbsql.generate_warnings	"0"	PHP_INI_SYSTEM
fbsql.autocommit	"1"	PHP_INI_SYSTEM

Name	Default	Changeable
fbsql.max_persistent	"-1"	PHP_INI_SYSTEM
fbsql.max_links	"128"	PHP_INI_SYSTEM
fbsql.max_connections	"128"	PHP_INI_SYSTEM
fbsql.max_results	"128"	PHP_INI_SYSTEM
fbsql.batchSize	"1000"	PHP_INI_SYSTEM
fbsql.default_host	NULL	PHP_INI_SYSTEM
fbsql.default_user	"_SYSTEM"	PHP_INI_SYSTEM
fbsql.default_password	""	PHP_INI_SYSTEM
fbsql.default_database	""	PHP_INI_SYSTEM
fbsql.default_database_password	""	PHP_INI_SYSTEM

For further details and definitions of the PHP_INI_* constants, see the Apéndice H.

Tipos de recursos

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

FBSQL_ASSOC (integer)

FBSQL_NUM (integer)

FBSQL_BOTH (integer)

FBSQL_LOCK_DEFERRED (integer)

FBSQL_LOCK_OPTIMISTIC (integer)

FBSQL_LOCK_PESSIMISTIC (integer)

FBSQL_ISO_READ_UNCOMMITTED (integer)

FBSQL_ISO_READ_COMMITTED (integer)

FBSQL_ISO_REPEATABLE_READ (integer)

FBSQL_ISO_SERIALIZABLE (integer)

FBSQL_ISO_VERSIONED (integer)

FBSQL_UNKNOWN (integer)

FBSQL_STOPPED (integer)

FBSQL_STARTING (integer)

FBSQL_RUNNING (integer)

FBSQL_STOPPING (integer)

FBSQL_NOEXEC (integer)

FBSQL_LOB_DIRECT (integer)

FBSQL_LOB_HANDLE (integer)

Tabla de contenidos

fbsql_affected_rows

(PHP 4 >= 4.0.6, PHP 5)

fbsql_affected_rows -- Get number of affected rows in previous FrontBase operation

Descripción

int **fbsql_affected_rows** ([resource link_identifier])

fbsql_affected_rows() returns the number of rows affected by the last INSERT, UPDATE or DELETE query associated with *link_identifier*. If the link identifier isn't specified, the last link opened by **fbsql_connect()** is assumed.

> **Nota:** If you are using transactions, you need to call **fbsql_affected_rows()** after your INSERT, UPDATE, or DELETE query, not after the commit.

If the last query was a DELETE query with no WHERE clause, all of the records will have been deleted from the table but this function will return zero.

> **Nota:** When using UPDATE, FrontBase will not update columns where the new value is the same as the old value. This creates the possibility that **fbsql_affected_rows()** may not actually equal the number of rows matched, only the number of rows that were literally affected by the query.

If the last query failed, this function will return -1.

See also: **fbsql_num_rows()**.

fbsql_autocommit

(PHP 4 >= 4.0.6, PHP 5)

fbsql_autocommit -- Enable or disable autocommit

Descripción

bool **fbsql_autocommit** (resource link_identifier [, bool OnOff])

fbsql_autocommit() returns the current autocommit status. If the optional OnOff parameter is given the auto commit status will be changed. With OnOff set to **TRUE** each statement will be committed automatically, if no errors was found. With OnOff set to **FALSE** the user must commit or rollback the transaction using either **fbsql_commit()** or **fbsql_rollback()**.

See also: **fbsql_commit()** and **fbsql_rollback()**

fbsql_blob_size

(PHP 4 >= 4.2.0, PHP 5)

fbsql_blob_size -- Get the size of a BLOB

Descripción

int **fbsql_blob_size** (string blob_handle [, resource link_identifier])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

fbsql_change_user

(no version information, might be only in CVS)

fbsql_change_user -- Change logged in user of the active connection

Descripción

resource **fbsql_change_user** (string user, string password [, string database [, resource link_identifier]])

fbsql_change_user() changes the logged in user of the current active connection, or the connection given by the optional parameter link_identifier. If a database is specified, this will default or current database after the user has been changed. If the new user and password authorization fails, the

current connected user stays active.

fbsql_clob_size

(PHP 4 >= 4.2.0, PHP 5)

fbsql_clob_size -- Get the size of a CLOB

Descripción

int **fbsql_clob_size** (string clob_handle [, resource link_identifier])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

fbsql_close

(PHP 4 >= 4.0.6, PHP 5)

fbsql_close -- Close FrontBase connection

Descripción

bool **fbsql_close** ([resource link_identifier])

Returns: **`TRUE`** on success, **`FALSE`** on error.

fbsql_close() closes the connection to the FrontBase server that's associated with the specified link identifier. If *link_identifier* isn't specified, the last opened link is used.

Using **fbsql_close()** isn't usually necessary, as non-persistent open links are automatically closed at the end of the script's execution.

Ejemplo 1. fbsql_close() example

```
<?php
    $link = fbsql_connect("localhost", "_SYSTEM", "secret")
        or die("Could not connect");
    echo "Connected successfully";
    fbsql_close($link);
?>
```

See also: **fbsql_connect()** and **fbsql_pconnect()**.

fbsql_commit

(PHP 4 >= 4.0.6, PHP 5)

fbsql_commit -- Commits a transaction to the database

Descripción

bool **fbsql_commit** ([resource link_identifier])

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

fbsql_commit() ends the current transaction by writing all inserts, updates and deletes to the disk and unlocking all row and table locks held by the transaction. This command is only needed if autocommit is set to false.

See also: **fbsql_autocommit()** and **fbsql_rollback()**

fbsql_connect

(PHP 4 >= 4.0.6, PHP 5)

fbsql_connect -- Open a connection to a FrontBase Server

Descripción

resource **fbsql_connect** ([string hostname [, string username [, string password]]])

Returns a positive FrontBase link identifier on success, or an error message on failure.

fbsql_connect() establishes a connection to a FrontBase server. The following defaults are assumed for missing optional parameters: *hostname* = '**NULL**', *username* = '_SYSTEM' and *password* = empty password.

If a second call is made to **fbsql_connect()** with the same arguments, no new link will be established, but instead, the link identifier of the already opened link will be returned.

The link to the server will be closed as soon as the execution of the script ends, unless it's closed earlier by explicitly calling **fbsql_close()**.

Ejemplo 1. fbsql_connect() example

```
<?php

    $link = fbsql_connect("localhost", "_SYSTEM", "secret")
        or die("Could not connect");
    echo "Connected successfully";
    fbsql_close($link);

?>
```

See also **fbsql_pconnect()** and **fbsql_close()**.

fbsql_create_blob

(PHP 4 >= 4.2.0, PHP 5)

fbsql_create_blob -- Create a BLOB

Descripción

string **fbsql_create_blob** (string blob_data [, resource link_identifier])

Returns: A resource handle to the newly created blob.

fbsql_create_blob() creates a blob from blob_data. The returned resource handle can be used with insert and update commands to store the blob in the database.

Ejemplo 1. fbsql_create_blob() example

```
<?php
    $link = fbsql_pconnect("localhost", "_SYSTEM", "secret")
        or die("Could not connect");
    $filename = "blobfile.bin";
    $fp = fopen($filename, "rb");
    $blobdata = fread($fp, filesize($filename));
    fclose($fp);

    $blobHandle = fbsql_create_blob($blobdata, $link);

    $sql = "INSERT INTO BLOB_TABLE (BLOB_COLUMN) VALUES ($blobHandle);";
    $rs = fbsql_query($sql, $link);
?>
```

See also: fbsql_create_clob(), fbsql_read_blob(), fbsql_read_clob(), and fbsql_set_lob_mode().

fbsql_create_clob

(PHP 4 >= 4.2.0, PHP 5)

fbsql_create_clob -- Create a CLOB

Descripción

string **fbsql_create_clob** (string clob_data [, resource link_identifier])

Returns: A resource handle to the newly created CLOB.

fbsql_create_clob() creates a clob from clob_data. The returned resource handle can be used with insert and update commands to store the clob in the database.

Ejemplo 1. fbsql_create_clob() example

```
<?php
    $link = fbsql_pconnect("localhost", "_SYSTEM", "secret")
        or die("Could not connect");
    $filename = "clob_file.txt";
    $fp = fopen($filename, "rb");
    $clobdata = fread($fp, filesize($filename));
    fclose($fp);

    $clobHandle = fbsql_create_clob($clobdata, $link);

    $sql = "INSERT INTO CLOB_TABLE (CLOB_COLUMN) VALUES ($clobHandle);";
    $rs = fbsql_query($sql, $link);
?>
```

See also: fbsql_create_blob(), fbsql_read_blob(), fbsql_read_clob(), and fbsql_set_lob_mode().

fbsql_create_db

(PHP 4 >= 4.0.6, PHP 5)

fbsql_create_db -- Create a FrontBase database

Descripción

bool **fbsql_create_db** (string database_name [, resource link_identifier [, string database_options]])

fbsql_create_db() attempts to create a new database named *database_name* on the server associated with the specified connection *link_identifier*.

Ejemplo 1. fbsql_create_db() example

```
<?php
    $link = fbsql_pconnect("localhost", "_SYSTEM", "secret")
        or die("Could not connect");
    if (fbsql_create_db("my_db")) {
        echo "Database created successfully\n";
    } else {
        printf("Error creating database: %s\n", fbsql_error());
    }
?>
```

See also: **fbsql_drop_db()**.

fbsql_data_seek

(PHP 4 >= 4.0.6, PHP 5)

fbsql_data_seek -- Move internal result pointer

Descripción

bool **fbsql_data_seek** (resource result_identifier, int row_number)

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

fbsql_data_seek() moves the internal row pointer of the FrontBase result associated with the specified result identifier to point to the specified row number. The next call to **fbsql_fetch_row()** would return that row.

Row_number starts at 0.

Ejemplo 1. fbsql_data_seek() example

```
<?php
    $link = fbsql_pconnect("localhost", "_SYSTEM", "secret")
        or die("Could not connect");

    fbsql_select_db("samp_db")
        or die("Could not select database");

    $query = "SELECT last_name, first_name FROM friends;";
    $result = fbsql_query($query)
        or die("Query failed");

    // fetch rows in reverse order

    for ($i = fbsql_num_rows($result) - 1; $i >=0; $i--) {
        if (!fbsql_data_seek($result, $i)) {
            printf("Cannot seek to row %d\n", $i);
            continue;
        }

        if (!($row = fbsql_fetch_object($result)))
            continue;

        echo $row->last_name . $row->first_name . "<br />\n";
    }

    fbsql_free_result($result);
?>
```

fbsql_database_password

(PHP 4 >= 4.0.6, PHP 5)

fbsql_database_password -- Sets or retrieves the password for a FrontBase database

Descripción

string **fbsql_database_password** (resource link_identifier [, string database_password])

Returns: The database password associated with the link identifier.

fbsql_database_password() sets and retrieves the database password used by the connection. if a database is protected by a database password, the user must call this function before calling **fbsql_select_db()**. if the second optional parameter is given the function sets the database password for the specified link identifier. If no link identifier is specified, the last opened link is assumed. If no link is open, the function will try to establish a link as if **fbsql_connect()** was called, and use it.

This function does not change the database password in the database nor can it be used to retrieve the database password for a database.

Ejemplo 1. fbsql_create_clob() example

```
<?php
    $link = fbsql_pconnect("localhost", "_SYSTEM", "secret")
        or die("Could not connect");
    fbsql_database_password($link, "secret db password");
    fbsql_select_db($database, $link);
?>
```

See also: **fbsql_connect()**, **fbsql_pconnect()** and **fbsql_select_db()**.

fbsql_database

(PHP 4 >= 4.0.6, PHP 5)

fbsql_database -- Get or set the database name used with a connection

Descripción

string **fbsql_database** (resource link_identifier [, string database])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

fbsql_db_query

(PHP 4 >= 4.0.6, PHP 5)

fbsql_db_query -- Send a FrontBase query

Descripción

resource **fbsql_db_query** (string database, string query [, resource link_identifier])

Returns: A positive FrontBase result identifier to the query result, or **`FALSE`** on error.

fbsql_db_query() selects a database and executes a query on it. If the optional link identifier isn't specified, the function will try to find an open link to the FrontBase server and if no such link is found it'll try to create one as if **fbsql_connect()** was called with no arguments

See also **fbsql_connect()**.

fbsql_db_status

(PHP 4 >= 4.1.0, PHP 5)

fbsql_db_status -- Get the status for a given database

Descripción

int **fbsql_db_status** (string database_name [, resource link_identifier])

Returns: An integer value with the current status.

fbsql_db_status() requests the current status of the database specified by *database_name*. If the *link_identifier* is omitted the default link_identifier will be used.

The return value can be one of the following constants:

- **FALSE** - The exec handler for the host was invalid. This error will occur when the link_identifier connects directly to a database by using a port number. FBExec can be available on the server but no connection has been made for it.
- FBSQL_UNKNOWN - The Status is unknown.
- FBSQL_STOPPED - The database is not running. Use **fbsql_start_db()** to start the database.
- FBSQL_STARTING - The database is starting.
- FBSQL_RUNNING - The database is running and can be used to perform SQL operations.
- FBSQL_STOPPING - The database is stopping.
- FBSQL_NOEXEC - FBExec is not running on the server and it is not possible to get the status of the database.

See also: **fbsql_start_db()** and **fbsql_stop_db()**.

fbsql_drop_db

(PHP 4 >= 4.0.6, PHP 5)

fbsql_drop_db -- Drop (delete) a FrontBase database

Descripción

bool **fbsql_drop_db** (string database_name [, resource link_identifier])

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

fbsql_drop_db() attempts to drop (remove) an entire database from the server associated with the specified link identifier.

fbsql_errno

(PHP 4 >= 4.0.6, PHP 5)

fbsql_errno -- Returns the numerical value of the error message from previous FrontBase operation

Descripción

int **fbsql_errno** ([resource link_identifier])

Returns the error number from the last fbsql function, or *0* (zero) if no error occurred.

Errors coming back from the fbsql database backend don't issue warnings. Instead, use **fbsql_errno()** to retrieve the error code. Note that this function only returns the error code from the most recently executed fbsql function (not including **fbsql_error()** and **fbsql_errno()**), so if you want to use it, make sure you check the value before calling another fbsql function.

```
<?php
fbsql_connect("marliesle");
echo fbsql_errno() . ": " . fbsql_error() . "<br />";
fbsql_select_db("nonexistentdb");
echo fbsql_errno() . ": " . fbsql_error() . "<br />";
$conn = fbsql_query("SELECT * FROM nonexistenttable;");
echo fbsql_errno() . ": " . fbsql_error() . "<br />";
?>
```

See also: **fbsql_error()** and **fbsql_warnings()**.

fbsql_error

(PHP 4 >= 4.0.6, PHP 5)

fbsql_error -- Returns the text of the error message from previous FrontBase operation

Descripción

string **fbsql_error** ([resource link_identifier])

Returns the error text from the last fbsql function, or '' (the empty string) if no error occurred.

Errors coming back from the fbsql database backend don't issue warnings. Instead, use **fbsql_error()** to retrieve the error text. Note that this function only returns the error text from the most recently executed fbsql function (not including **fbsql_error()** and **fbsql_errno()**), so if you want to use it, make sure you check the value before calling another fbsql function.

```
<?php
fbsql_connect("marliesle");
echo fbsql_errno() . ": " . fbsql_error() . "<br />";
fbsql_select_db("nonexistentdb");
echo fbsql_errno() . ": " . fbsql_error() . "<br />";
$conn = fbsql_query("SELECT * FROM nonexistenttable;");
echo fbsql_errno() . ": " . fbsql_error() . "<br />";
?>
```

See also: **fbsql_errno()** and **fbsql_warnings()**.

fbsql_fetch_array

(PHP 4 >= 4.0.6, PHP 5)

fbsql_fetch_array -- Fetch a result row as an associative array, a numeric array, or both

Descripción

array **fbsql_fetch_array** (resource result [, int result_type])

Returns an array that corresponds to the fetched row, or **FALSE** if there are no more rows.

fbsql_fetch_array() is an extended version of **fbsql_fetch_row()**. In addition to storing the data in the numeric indices of the result array, it also stores the data in associative indices, using the field names as keys.

If two or more columns of the result have the same field names, the last column will take precedence. To access the other column(s) of the same name, you must the numeric index of the column or make an alias for the column.

```
select t1.f1 as foo t2.f1 as bar from t1, t2
```

An important thing to note is that using **fbsql_fetch_array()** is NOT significantly slower than using **fbsql_fetch_row()**, while it provides a significant added value.

The optional second argument *result_type* in **fbsql_fetch_array()** is a constant and can take the following values: FBSQL_ASSOC, FBSQL_NUM, and FBSQL_BOTH.

For further details, see also **fbsql_fetch_row()** and **fbsql_fetch_assoc()**.

Ejemplo 1. fbsql_fetch_array() example

```
<?php
fbsql_connect($host, $user, $password);
$result = fbsql_db_query("database", "select user_id, fullname from table");
while ($row = fbsql_fetch_array($result)) {
    echo "user_id: " . $row["user_id"] . "<br />\n";
    echo "user_id: " . $row[0] . "<br />\n";
    echo "fullname: " . $row["fullname"] . "<br />\n";
    echo "fullname: " . $row[1] . "<br />\n";
}
fbsql_free_result($result);
?>
```

fbsql_fetch_assoc

(PHP 4 >= 4.0.6, PHP 5)

fbsql_fetch_assoc -- Fetch a result row as an associative array

Descripción

array **fbsql_fetch_assoc** (resource result)

Returns an associative array that corresponds to the fetched row, or **FALSE** if there are no more rows.

fbsql_fetch_assoc() is ñalent to calling **fbsql_fetch_array()** with FBSQL_ASSOC for the optional second parameter. It only returns an associative array. This is the way **fbsql_fetch_array()** originally worked. If you need the numeric indices as well as the associative, use **fbsql_fetch_array()**.

If two or more columns of the result have the same field names, the last column will take precedence. To access the other column(s) of the same name, you must use **fbsql_fetch_array()** and have it return the numeric indices as well.

An important thing to note is that using **fbsql_fetch_assoc()** is NOT significantly slower than using **fbsql_fetch_row()**, while it provides a significant added value.

For further details, see also **fbsql_fetch_row()** and **fbsql_fetch_array()**.

Ejemplo 1. fbsql_fetch_assoc() example

```
<?php
fbsql_connect($host, $user, $password);
$result = fbsql_db_query("database", "select * from table");
while ($row = fbsql_fetch_assoc($result)) {
    echo $row["user_id"];
    echo $row["fullname"];
}
fbsql_free_result($result);
?>
```

fbsql_fetch_field

(PHP 4 >= 4.0.6, PHP 5)

fbsql_fetch_field -- Get column information from a result and return as an object

Descripción

object **fbsql_fetch_field** (resource result [, int field_offset])

Returns an object containing field information.

fbsql_fetch_field() can be used in order to obtain information about fields in a certain query result. If the field offset isn't specified, the next field that wasn't yet retrieved by **fbsql_fetch_field()** is retrieved.

The properties of the object are:

- name - column name
- table - name of the table the column belongs to
- max_length - maximum length of the column
- not_null - 1 if the column cannot be **`NULL`**
- type - the type of the column

Ejemplo 1. fbsql_fetch_field() example

```
<?php
fbsql_connect($host, $user, $password)
    or die("Could not connect");
$result = fbsql_db_query("database", "select * from table")
    or die("Query failed");
# get column metadata
$i = 0;
while ($i < fbsql_num_fields($result)) {
    echo "Information for column $i:<br />\n";
    $meta = fbsql_fetch_field($result);
    if (!$meta) {
        echo "No information available<br />\n";
    }
    echo "<pre>
max_length:   $meta->max_length
name:         $meta->name
not_null:     $meta->not_null
table:        $meta->table
type:         $meta->type
</pre>";
    $i++;
}
fbsql_free_result($result);
?>
```

See also **fbsql_field_seek()**.

fbsql_fetch_lengths

(PHP 4 >= 4.0.6, PHP 5)

fbsql_fetch_lengths -- Get the length of each output in a result

Descripción

array **fbsql_fetch_lengths** (resource result)

Returns: An array that corresponds to the lengths of each field in the last row fetched by **fbsql_fetch_row()**, or **FALSE** on error.

fbsql_fetch_lengths() stores the lengths of each result column in the last row returned by **fbsql_fetch_row()**, **fbsql_fetch_array()** and **fbsql_fetch_object()** in an array, starting at offset 0.

See also: **fbsql_fetch_row()**.

fbsql_fetch_object

(PHP 4 >= 4.0.6, PHP 5)

fbsql_fetch_object -- Fetch a result row as an object

Descripción

object **fbsql_fetch_object** (resource result [, int result_type])

Returns an object with properties that correspond to the fetched row, or **FALSE** if there are no more rows.

fbsql_fetch_object() is similar to **fbsql_fetch_array()**, with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

The optional argument *result_type* is a constant and can take the following values: FBSQL_ASSOC, FBSQL_NUM, and FBSQL_BOTH.

Speed-wise, the function is identical to **fbsql_fetch_array()**, and almost as quick as **fbsql_fetch_row()** (the difference is insignificant).

Ejemplo 1. fbsql_fetch_object() example

```
<?php
fbsql_connect($host, $user, $password);
$result = fbsql_db_query("database", "select * from table");
while ($row = fbsql_fetch_object($result)) {
    echo $row->user_id;
    echo $row->fullname;
}
fbsql_free_result($result);
?>
```

See also: **fbsql_fetch_array()** and **fbsql_fetch_row()**.

fbsql_fetch_row

(PHP 4 >= 4.0.6, PHP 5)

fbsql_fetch_row -- Get a result row as an enumerated array

Descripción

array **fbsql_fetch_row** (resource result)

Returns: An array that corresponds to the fetched row, or **FALSE** if there are no more rows.

fbsql_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0.

Subsequent call to **fbsql_fetch_row()** would return the next row in the result set, or **FALSE** if there are no more rows.

See also: **fbsql_fetch_array()**, **fbsql_fetch_object()**, **fbsql_data_seek()**, **fbsql_fetch_lengths()**, and **fbsql_result()**.

fbsql_field_flags

(PHP 4 >= 4.0.6, PHP 5)

fbsql_field_flags -- Get the flags associated with the specified field in a result

Descripción

string **fbsql_field_flags** (resource result [, int field_offset])

fbsql_field_flags() returns the field flags of the specified field. The flags are reported as a single word per flag separated by a single space, so that you can split the returned value using **explode()**.

fbsql_field_len

(PHP 4 >= 4.0.6, PHP 5)

fbsql_field_len -- Returns the length of the specified field

Descripción

int **fbsql_field_len** (resource result [, int field_offset])

fbsql_field_len() returns the length of the specified field.

fbsql_field_name

(PHP 4 >= 4.0.6, PHP 5)

fbsql_field_name -- Get the name of the specified field in a result

Descripción

string **fbsql_field_name** (resource result [, int field_index])

fbsql_field_name() returns the name of the specified field index. *result* must be a valid result identifier and *field_index* is the numerical offset of the field.

> **Nota:** *field_index* starts at 0.
>
> e.g. The index of the third field would actually be 2, the index of the fourth field would be 3 and so on.

Ejemplo 1. fbsql_field_name() example

```
<?php
// The users table consists of three fields:
//   user_id
//   username
//   password.

$res = fbsql_db_query("users", "select * from users", $link);

echo fbsql_field_name($res, 0) . "\n";
echo fbsql_field_name($res, 2);
?>
```

El resultado del ejemplo seria:

```
user_id
password
```

fbsql_field_seek

(PHP 4 >= 4.0.6, PHP 5)

fbsql_field_seek -- Set result pointer to a specified field offset

Descripción

bool **fbsql_field_seek** (resource result [, int field_offset])

Seeks to the specified field offset. If the next call to **fbsql_fetch_field()** doesn't include a field offset, the field offset specified in **fbsql_field_seek()** will be returned.

See also: **fbsql_fetch_field()**.

fbsql_field_table

(PHP 4 >= 4.0.6, PHP 5)

fbsql_field_table -- Get name of the table the specified field is in

Descripción

string **fbsql_field_table** (resource result [, int field_offset])

Returns the name of the table that the specified field is in.

fbsql_field_type

(PHP 4 >= 4.0.6, PHP 5)

fbsql_field_type -- Get the type of the specified field in a result

Descripción

string **fbsql_field_type** (resource result [, int field_offset])

fbsql_field_type() is similar to the **fbsql_field_name()** function. The arguments are identical, but the field type is returned instead. The field type will be one of "int", "real", "string", "blob", and others as detailed in the FrontBase documentation.

Ejemplo 1. fbsql_field_type() example

```php
<?php

fbsql_connect("localhost", "_SYSTEM", "");
fbsql_select_db("wisconsin");
$result = fbsql_query("SELECT * FROM onek;");
$fields = fbsql_num_fields($result);
$rows   = fbsql_num_rows($result);
$i = 0;
$table = fbsql_field_table($result, $i);
echo "Your '" . $table . "' table has " . $fields . " fields and " . $rows . " records
echo "The table has the following fields <br />";
while ($i < $fields) {
    $type  = fbsql_field_type($result, $i);
    $name  = fbsql_field_name($result, $i);
    $len   = fbsql_field_len($result, $i);
    $flags = fbsql_field_flags($result, $i);
    echo $type . " " . $name . " " . $len . " " . $flags . "<br />";
    $i++;
}
fbsql_close();

?>
```

fbsql_free_result

(PHP 4 >= 4.0.6, PHP 5)

fbsql_free_result -- Free result memory

Descripción

bool **fbsql_free_result** (resource result)

fbsql_free_result() will free all memory associated with the result identifier *result*.

fbsql_free_result() only needs to be called if you are concerned about how much memory is being used for queries that return large result sets. All associated result memory is automatically freed at the end of the script's execution.

fbsql_get_autostart_info

(PHP 4 >= 4.1.0, PHP 5)

fbsql_get_autostart_info -- No description given yet

Descripción

array **fbsql_get_autostart_info** ([resource link_identifier])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

fbsql_hostname

(PHP 4 >= 4.0.6, PHP 5)

fbsql_hostname -- Get or set the host name used with a connection

Descripción

string **fbsql_hostname** (resource link_identifier [, string host_name])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

fbsql_insert_id

(PHP 4 >= 4.0.6, PHP 5)

fbsql_insert_id -- Get the id generated from the previous INSERT operation

Descripción

int **fbsql_insert_id** ([resource link_identifier])

fbsql_insert_id() returns the ID generated for an column defined as DEFAULT UNIQUE by the previous INSERT query using the given *link_identifier*. If *link_identifier* isn't specified, the last opened link is assumed.

fbsql_insert_id() returns 0 if the previous query does not generate an DEFAULT UNIQUE value. If you need to save the value for later, be sure to call fbsql_insert_id() immediately after the query that generates the value.

> **Nota:** The value of the FrontBase SQL function **fbsql_insert_id()** always contains the most recently generated DEFAULT UNIQUE value, and is not reset between queries.

fbsql_list_dbs

(PHP 4 >= 4.0.6, PHP 5)

fbsql_list_dbs -- List databases available on a FrontBase server

Descripción

resource **fbsql_list_dbs** ([resource link_identifier])

fbsql_list_dbs() will return a result pointer containing the databases available from the current fbsql daemon. Use the **fbsql_tablename()** function to traverse this result pointer.

Ejemplo 1. fbsql_list_dbs() example

```
$link = fbsql_connect('localhost', 'myname', 'secret');
$db_list = fbsql_list_dbs($link);

while ($row = fbsql_fetch_object($db_list)) {
    echo $row->Database . "\n";
}
```

El resultado del ejemplo seria:

```
database1
database2
database3
...
```

> **Nota:** The above code would just as easily work with **fbsql_fetch_row()** or other similar functions.

fbsql_list_fields

(PHP 4 >= 4.0.6, PHP 5)

fbsql_list_fields -- List FrontBase result fields

Descripción

resource **fbsql_list_fields** (string database_name, string table_name [, resource link_identifier])

fbsql_list_fields() retrieves information about the given tablename. Arguments are the database name and the table name. A result pointer is returned which can be used with **fbsql_field_flags()**, **fbsql_field_len()**, **fbsql_field_name()**, and **fbsql_field_type()**.

A result identifier is a positive integer. The function returns **FALSE** if an error occurs. A string describing the error will be placed in *$phperrmsg*, and unless the function was called as *@fbsql()* then this error string will also be printed out.

Ejemplo 1. fbsql_list_fields() example

```
<?php
$link = fbsql_connect('localhost', 'myname', 'secret');

$fields = fbsql_list_fields("database1", "table1", $link);
$columns = fbsql_num_fields($fields);

for ($i = 0; $i < $columns; $i++) {
    echo fbsql_field_name($fields, $i) . "\n";;
}
?>
```

El resultado del ejemplo seria:

```
field1
field2
field3
...
```

fbsql_list_tables

(PHP 4 >= 4.0.6, PHP 5)

fbsql_list_tables -- List tables in a FrontBase database

Descripción

resource **fbsql_list_tables** (string database [, resource link_identifier])

fbsql_list_tables() takes a database name and returns a result pointer much like the fbsql_db_query() function. The fbsql_tablename() function should be used to extract the actual table names from the result pointer.

fbsql_next_result

(PHP 4 >= 4.0.6, PHP 5)

fbsql_next_result -- Move the internal result pointer to the next result

Descripción

bool **fbsql_next_result** (resource result_id)

When sending more than one SQL statement to the server or executing a stored procedure with multiple results will cause the server to return multiple result sets. This function will test for additional results available form the server. If an additional result set exists it will free the existing result set and prepare to fetch the words from the new result set. The function will return **TRUE** if an additional result set was available or **FALSE** otherwise.

Ejemplo 1. fbsql_next_result() example

```
<?php
    $link = fbsql_connect("localhost", "_SYSTEM", "secret");
    fbsql_select_db("MyDB", $link);
    $SQL = "Select * from table1; select * from table2;";
    $rs = fbsql_query($SQL, $link);
    do {
        while ($row = fbsql_fetch_row($rs)) {
        }
    } while (fbsql_next_result($rs));
    fbsql_free_result($rs);
    fbsql_close($link);
?>
```

fbsql_num_fields

(PHP 4 >= 4.0.6, PHP 5)

fbsql_num_fields -- Get number of fields in result

Descripción

int **fbsql_num_fields** (resource result)

fbsql_num_fields() returns the number of fields in a result set.

See also: fbsql_db_query(), fbsql_query(), fbsql_fetch_field(), and fbsql_num_rows().

fbsql_num_rows

(PHP 4 >= 4.0.6, PHP 5)

fbsql_num_rows -- Get number of rows in result

Descripción

int **fbsql_num_rows** (resource result)

fbsql_num_rows() returns the number of rows in a result set. This command is only valid for SELECT statements. To retrieve the number of rows returned from a INSERT, UPDATE or DELETE query, use **fbsql_affected_rows()**.

Ejemplo 1. fbsql_num_rows() example

```
<?php

$link = fbsql_connect("localhost", "username", "password");
fbsql_select_db("database", $link);

$result = fbsql_query("SELECT * FROM table1;", $link);
$num_rows = fbsql_num_rows($result);

echo "$num_rows Rows\n";

?>
```

See also: **fbsql_affected_rows()**, **fbsql_connect()**, **fbsql_select_db()**, and **fbsql_query()**.

fbsql_password

(PHP 4 >= 4.0.6, PHP 5)

fbsql_password -- Get or set the user password used with a connection

Descripción

string **fbsql_password** (resource link_identifier [, string password])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

fbsql_pconnect

(PHP 4 >= 4.0.6, PHP 5)

fbsql_pconnect -- Open a persistent connection to a FrontBase Server

Descripción

resource **fbsql_pconnect** ([string hostname [, string username [, string password]]])

Returns: A positive FrontBase persistent link identifier on success, or **FALSE** on error.

fbsql_pconnect() establishes a connection to a FrontBase server. The following defaults are assumed for missing optional parameters: *host* = 'localhost', *username* = "_SYSTEM" and *password* = empty password.

To set Frontbase server port number, use **fbsql_select_db()**.

fbsql_pconnect() acts very much like **fbsql_connect()** with two major differences.

First, when connecting, the function would first try to find a (persistent) link that's already open with the same host, username and password. If one is found, an identifier for it will be returned instead of opening a new connection.

Second, the connection to the SQL server will not be closed when the execution of the script ends. Instead, the link will remain open for future use.

This type of links is therefore called 'persistent'.

fbsql_query

(PHP 4 >= 4.0.6, PHP 5)

fbsql_query -- Send a FrontBase query

Descripción

resource **fbsql_query** (string query [, resource link_identifier [, int batch_size]])

fbsql_query() sends a query to the currently active database on the server that's associated with the specified link identifier. If *link_identifier* isn't specified, the last opened link is assumed. If no link is open, the function tries to establish a link as if **fbsql_connect()** was called with no arguments, and use it.

> **Nota:** The query string shall always end with a semicolon.

fbsql_query() returns **TRUE** (non-zero) or **FALSE** to indicate whether or not the query succeeded. A return value of **TRUE** means that the query was legal and could be executed by the server. It does not indicate anything about the number of rows affected or returned. It is perfectly possible for a query to succeed but affect no rows or return no rows.

The following query is syntactically invalid, so **fbsql_query()** fails and returns **FALSE**:
Ejemplo 1. fbsql_query() example

```
<?php
$result = fbsql_query("SELECT * WHERE 1=1")
    or die ("Invalid query");
?>
```

The following query is semantically invalid if *my_col* is not a column in the table *my_tbl*, so **fbsql_query()** fails and returns **FALSE**:
Ejemplo 2. fbsql_query() example

```
<?php
$result = fbsql_query ("SELECT my_col FROM my_tbl")
    or die ("Invalid query");
?>
```

fbsql_query() will also fail and return **FALSE** if you don't have permission to access the table(s) referenced by the query.

Assuming the query succeeds, you can call **fbsql_num_rows()** to find out how many rows were returned for a SELECT statement or **fbsql_affected_rows()** to find out how many rows were affected by a DELETE, INSERT, REPLACE, or UPDATE statement.

For SELECT statements, **fbsql_query()** returns a new result identifier that you can pass to **fbsql_result()**. When you are done with the result set, you can free the resources associated with it by calling **fbsql_free_result()**. Although, the memory will automatically be freed at the end of the script's execution.

See also: **fbsql_affected_rows()**, **fbsql_db_query()**, **fbsql_free_result()**, **fbsql_result()**, **fbsql_select_db()**, and **fbsql_connect()**.

fbsql_read_blob

(PHP 4 >= 4.2.0, PHP 5)

fbsql_read_blob -- Read a BLOB from the database

Descripción

string **fbsql_read_blob** (string blob_handle [, resource link_identifier])

Returns: A string containing the BLOB specified by blob_handle.

fbsql_read_blob() reads BLOB data from the database. If a select statement contains BLOB and/or CLOB columns FrontBase will return the data directly when data is fetched. This default behavior can be changed with **fbsql_set_lob_mode()** so the fetch functions will return handles to BLOB and CLOB data. If a handle is fetched a user must call **fbsql_read_blob()** to get the actual BLOB data from the database.

Ejemplo 1. fbsql_read_blob() example

```
<?php
    $link = fbsql_pconnect("localhost", "_SYSTEM", "secret")
        or die("Could not connect");
    $sql = "SELECT BLOB_COLUMN FROM BLOB_TABLE;";
    $rs = fbsql_query($sql, $link);
    $row_data = fbsql_fetch_row($rs);
    // $row_data[0] will now contain the blob data for the first row
    fbsql_free_result($rs);

    $rs = fbsql_query($sql, $link);
    fbsql_set_lob_mode($rs, FBSQL_LOB_HANDLE);
    $row_data = fbsql_fetch_row($rs);
    // $row_data[0] will now contain a handle to the BLOB data in the first row
    $blob_data = fbsql_read_blob($row_data[0]);
    fbsql_free_result($rs);

?>
```

See also: **fbsql_create_blob()**, **fbsql_read_blob()**, **fbsql_read_clob()**, and **fbsql_set_lob_mode()**.

fbsql_read_clob

(PHP 4 >= 4.2.0, PHP 5)

fbsql_read_clob -- Read a CLOB from the database

Descripción

string **fbsql_read_clob** (string clob_handle [, resource link_identifier])

Returns: A string containing the CLOB specified by clob_handle.

fbsql_read_clob() reads CLOB data from the database. If a select statement contains BLOB and/or CLOB columns FrontBase will return the data directly when data is fetched. This default behavior can be changed with **fbsql_set_lob_mode()** so the fetch functions will return handles to BLOB and CLOB data. If a handle is fetched a user must call **fbsql_read_clob()** to get the actual CLOB data from the database.

Ejemplo 1. fbsql_read_clob() example

```
<?php
    $link = fbsql_pconnect("localhost", "_SYSTEM", "secret")
        or die("Could not connect");
    $sql = "SELECT CLOB_COLUMN FROM CLOB_TABLE;";
    $rs = fbsql_query($sql, $link);
    $row_data = fbsql_fetch_row($rs);
    // $row_data[0] will now contain the clob data for the first row
    fbsql_free_result($rs);

    $rs = fbsql_query($sql, $link);
    fbsql_set_lob_mode($rs, FBSQL_LOB_HANDLE);
    $row_data = fbsql_fetch_row($rs);
    // $row_data[0] will now contain a handle to the CLOB data in the first row
    $clob_data = fbsql_read_clob($row_data[0]);
    fbsql_free_result($rs);

?>
```

See also: **fbsql_create_blob()**, **fbsql_read_blob()**, **fbsql_read_clob()**, and **fbsql_set_lob_mode()**.

fbsql_result

(PHP 4 >= 4.0.6, PHP 5)

fbsql_result -- Get result data

Descripción

mixed **fbsql_result** (resource result [, int row [, mixed field]])

fbsql_result() returns the contents of one cell from a FrontBase result set. The field argument can be the field's offset, or the field's name, or the field's table dot field's name (tabledname.fieldname). If the column name has been aliased ('select foo as bar from...'), use the alias instead of the column name.

When working on large result sets, you should consider using one of the functions that fetch an

entire row (specified below). As these functions return the contents of multiple cells in one function call, they're MUCH quicker than **fbsql_result()**. Also, note that specifying a numeric offset for the field argument is much quicker than specifying a fieldname or tablename.fieldname argument.

Calls to **fbsql_result()** should not be mixed with calls to other functions that deal with the result set.

Recommended high-performance alternatives: **fbsql_fetch_row()**, **fbsql_fetch_array()**, and **fbsql_fetch_object()**.

fbsql_rollback

(PHP 4 >= 4.0.6, PHP 5)

fbsql_rollback -- Rollback a transaction to the database

Descripción

bool **fbsql_rollback** ([resource link_identifier])

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

fbsql_rollback() ends the current transaction by rolling back all statements issued since last commit. This command is only needed if autocommit is set to false.

See also: **fbsql_autocommit()** and **fbsql_commit()**

fbsql_select_db

(PHP 4 >= 4.0.6, PHP 5)

fbsql_select_db -- Select a FrontBase database

Descripción

bool **fbsql_select_db** ([string database_name [, resource link_identifier]])

fbsql_select_db() sets the current active database on the server that's associated with the specified link identifier. If no link identifier is specified, the last opened link is assumed. If no link is open, the function will try to establish a link as if **fbsql_connect()** was called, and use it.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

The client contacts FBExec to obtain the port number to use for the connection to the database. If the database name is a number the system will use that as a port number and it will not ask FBExec for the port number. The FrontBase server can be stared as FRontBase -FBExec=No -port=<port number> <database name>.

Every subsequent call to **fbsql_query()** will be made on the active database.

if the database is protected with a database password, the user must call **fbsql_database_password()** before selecting the database.

See also **fbsql_connect()**, **fbsql_pconnect()**, **fbsql_database_password()**, and **fbsql_query()**.

fbsql_set_lob_mode

(PHP 4 >= 4.2.0, PHP 5)

fbsql_set_lob_mode -- Set the LOB retrieve mode for a FrontBase result set

Descripción

bool **fbsql_set_lob_mode** (resource result, string database_name)

Returns: **TRUE** on success, **FALSE** on error.

fbsql_set_lob_mode() sets the mode for retrieving LOB data from the database. When BLOB and CLOB data is stored in FrontBase it can be stored direct or indirect. Direct stored LOB data will always be fetched no matter the setting of the lob mode. If the LOB data is less than 512 bytes it will always be stored directly.

- FBSQL_LOB_DIRECT - LOB data is retrieved directly. When data is fetched from the database with **fbsql_fetch_row()**, and other fetch functions, all CLOB and BLOB columns will be returned as ordinary columns. This is the default value on a new FrontBase result.
- FBSQL_LOB_HANDLE - LOB data is retrieved as handles to the data. When data is fetched from the database with **fbsql_fetch_row ()**, and other fetch functions, LOB data will be returned as a handle to the data if the data is stored indirect or the data if it is stored direct. If a handle is returned it will be a 27 byte string formatted as "@'00000000000000000000000'".

See also: **fbsql_create_blob()**, **fbsql_create_clob()**, **fbsql_read_blob()**, and **fbsql_read_clob()**.

fbsql_set_password

(PHP 5)

fbsql_set_password -- Change the password for a given user

Descripción

bool **fbsql_set_password** (resource link_identifier, string user, string password, string old_password)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

fbsql_set_transaction

(PHP 4 >= 4.2.0, PHP 5)

fbsql_set_transaction -- Set the transaction locking and isolation

Descripción

void **fbsql_set_transaction** (resource link_identifier, int Locking, int Isolation)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

fbsql_start_db

(PHP 4 >= 4.0.6, PHP 5)

fbsql_start_db -- Start a database on local or remote server

Descripción

bool **fbsql_start_db** (string database_name [, resource link_identifier [, string database_options]])

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

fbsql_start_db()

See also: **fbsql_db_status()** and **fbsql_stop_db()**.

fbsql_stop_db

(PHP 4 >= 4.0.6, PHP 5)

fbsql_stop_db -- Stop a database on local or remote server

Descripción

bool **fbsql_stop_db** (string database_name [, resource link_identifier])

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

fbsql_stop_db()

See also: **fbsql_db_status()** and **fbsql_start_db()**.

fbsql_tablename

(PHP 4 >= 4.2.0, PHP 5)

fbsql_tablename -- Get table name of field

Descripción

string **fbsql_tablename** (resource result, int i)

fbsql_tablename() takes a result pointer returned by the **fbsql_list_tables()** function as well as an integer index and returns the name of a table. The **fbsql_num_rows()** function may be used to determine the number of tables in the result pointer.

Ejemplo 1. fbsql_tablename() example

```
<?php
fbsql_connect("localhost", "_SYSTEM", "");
$result = fbsql_list_tables("wisconsin");
$i = 0;
while ($i < fbsql_num_rows($result)) {
    $tb_names[$i] = fbsql_tablename($result, $i);
    echo $tb_names[$i] . "<br />";
    $i++;
}
?>
```

fbsql_username

(PHP 4 >= 4.0.6, PHP 5)

fbsql_username -- Get or set the host user used with a connection

Descripción

string **fbsql_username** (resource link_identifier [, string username])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

fbsql_warnings

(PHP 4 >= 4.0.6, PHP 5)

fbsql_warnings -- Enable or disable FrontBase warnings

Descripción

bool **fbsql_warnings** ([bool OnOff])

Returns **TRUE** if warnings is turned on otherwise **FALSE**.

fbsql_warnings() enables or disables FrontBase warnings.

XXXV. Funciones del Formato de Datos de Formulario

Introducción

El Formato de Datos de Formulario (FDF) es un formato para la gestión de formularios al interior de documentos PDF. Es recomendable que lea la documentación en http://partners.adobe.com/asn/acrobat/forms.jsp para más información sobre lo que FDF es, y el modo de usarlo en general.

La idea general de FDF es similar a la de los formularios HTML. La diferencia es básicamente el formato en que los datos son transmitidos al servidor cuando el botón de envío es pulsado (el cual viene a ser el Formato de Datos de Formulario) y el formato del formulario mismo (el cual es el Formato de Documento Portable, PDF). El procesamiento de los datos FDF es una de las características ofrecidas por las funciones fdf. Pero hay más. También es posible tomar un formulario PDF existente y poblar los campos de entrada con datos sin modificar el formulario mismo. En tal caso, es posible crear un documento FDF (**fdf_create()**), definir los valores de cada campo de entrada (**fdf_set_value()**) y asociarlo con un formulario PDF (**fdf_set_file()**). Finalmente debe ser enviado al navegador con el tipo Mime *application/vnd.fdf*. El plugin lector de Acrobat en su navegador reconoce el tipo Mime, lee el formulario PDF asociado y aplica los datos del documento FDF.

Si echa un vistazo a un documento FDF con un editor de texto, encontrará un objeto de catálogo con el nombre *FDF*. Tal objeto puede contener cierto número de entradas como *Fields*, *F*, *Status* etc.. Las entradas usadas con más frecuencia son *Fields*, la cual apunta a una lista de campos de entrada, y *F* que contiene el nombre de archivo del documento PDF al que pertenecen estos datos. Esas entradas son referenciadas en la documentación FDF como /F-Key o /Status-Key. La modificación de estas entradas se realiza por medio de funciones como **fdf_set_file()** y **fdf_set_status()**. Los campos son mofidicados con **fdf_set_value()**, **fdf_set_opt()** etc..

Requirimientos

Es necesario el toolkit SDK para FDF disponible en http://partners.adobe.com/asn/acrobat/forms.jsp. A partir de PHP 4.3 necesita por lo menos la versión 5.0 del SDK. La biblioteca del toolkit FDF se encuentra disponible en forma binaria únicamente, las plataformas soportadas por Adobe son Win32, Linux, Solaris y AIX.

Instalación

You must compile PHP with *--with-fdftk[=DIR]*.

> **Nota:** If you run into problems configuring PHP with fdftk support, check whether the header file `fdftk.h` and the library `libfdftk.so` are at the right place. The configure script supports both the directory structure of the FDF SDK distribution and the usual `DIR/include` / `DIR/lib` layout, so you can point it either directly to the unpacked distribution directory or put the header file and the appropriate library for your platform into e.g. `/usr/local/include` and `/usr/local/lib` and configure with *--with-fdftk=/usr/local*.

Note to Win32 Users: In order to enable this module on a Windows environment, you must copy `fdftk.dll` from the DLL folder of the PHP/Win32 binary package to the SYSTEM32 folder of your windows machine. (Ex: `C:\WINNT\SYSTEM32` or `C:\WINDOWS\SYSTEM32`)

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

fdf

La mayoría de funciones fdf requieren un recurso *fdf* como su primer parámetro. Un recurso *fdf* es un gestor con un archivo fdf abierto. Los recursos *fdf* pueden obtenerse usando **fdf_create()**, **fdf_open()** y **fdf_open_string()**.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

`FDFValue` (**integer**)

`FDFStatus` (**integer**)

`FDFFile` (**integer**)

`FDFID` (**integer**)

`FDFFf` (**integer**)

`FDFSetFf` (**integer**)

`FDFClearFf` (**integer**)

`FDFFlags` (**integer**)

`FDFSetF` (**integer**)

`FDFClrF` (**integer**)

`FDFAP` (**integer**)

`FDFAS` (**integer**)

FDFAction (integer)

FDFAA (integer)

FDFAPRef (integer)

FDFIF (integer)

FDFEnter (integer)

FDFExit (integer)

FDFDown (integer)

FDFUp (integer)

FDFFormat (integer)

FDFValidate (integer)

FDFKeystroke (integer)

FDFCalculate (integer)

FDFNormalAP (integer)

FDFRolloverAP (integer)

FDFDownAP (integer)

Ejemplos

Los siguientes ejemplos muestran únicamente la evaluación de datos del formulario.
Ejemplo 1. Evaluación de un documento FDF

```php
<?php
// Abrir un fdf desde la cadena de entrada entregara por la extension
// El formulario pdf contenia varios campos de entrada de texto con los
// nombres volumen, fecha, comentario, editorial, editor, y dos
// cuadros de verificacion mostrar_editorial y mostrar_editor.
$fdf = fdf_open_string($HTTP_FDF_DATA);
$volumen = fdf_get_value($fdf, "volumen");
echo "El campo volumen tiene el valor '<b>$volumen</b>'<br />";

$fecha = fdf_get_value($fdf, "fecha");
echo "El campo fecha tiene el valor '<b>$fecha</b>'<br />";

$comentario = fdf_get_value($fdf, "comentario");
echo "El campo comentario tiene el valor '<b>$comentario</b>'<br />";

if (fdf_get_value($fdf, "mostrar_editorial") == "On") {
  $editorial = fdf_get_value($fdf, "editorial");
  echo "El campo editorial tiene el valor '<b>$editorial</b>'<br />";
} else
  echo "La editorial no ser&aacute; mostrada.<br />";

if (fdf_get_value($fdf, "mostrar_editor") == "On") {
  $editor = fdf_get_value($fdf, "editor");
  echo "El campo editor tiene el valor '<b>$editor</b>'<br />";
} else
  echo "El editor no ser&aacute; mostrado.<br />";
fdf_close($fdf);
?>
```

Tabla de contenidos

fdf_set_submit_form_action -- Sets a submit form action of a field
fdf_set_target_frame -- Set target frame for form display
fdf_set_value -- Fijar el valor de un campo
fdf_set_version -- Sets version number for a FDF file

fdf_add_doc_javascript

(PHP 4 >= 4.3.0, PHP 5)

fdf_add_doc_javascript -- Adds javascript code to the FDF document

Description

bool **fdf_add_doc_javascript** (resource fdfdoc, string script_name, string script_code)

Adds a script to the FDF, which Acrobat then adds to the doc-level scripts of a document, once the FDF is imported into it. It is strongly suggested to use '\r' for linebreaks within *script_code*.

Ejemplo 1. Adding JavaScript code to a FDF

```
<?php
$fdf = fdf_create();
fdf_add_doc_javascript($fdf, "PlusOne", "function PlusOne(x)\r{\r  return x+1;\r}\r");
fdf_save($fdf);
?>
```

will create a FDF like this:

```
%FDF-1.2
%ÃƒÂ¢Ã‚Ã£Ã‚Ã  Ãƒâ€œ
1 0 obj
<<
/FDF << /JavaScript << /Doc [ (PlusOne)(function PlusOne\(x\)\r{\r  return x+1;\r}\r)]
>>
endobj
trailer
<<
/Root 1 0 R

>>
%%EOF
```

fdf_add_template

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

fdf_add_template -- Adds a template into the FDF document

Description

bool **fdf_add_template** (resource fdfdoc, int newpage, string filename, string template, int rename)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

fdf_close

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

fdf_close -- Cerrar un documento FDF

Descripción

void **fdf_close** (int fdf_document)

La función **fdf_close()** cierra un documento FDF.

Vea también **fdf_open()**.

fdf_create

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

fdf_create -- Crear un documento FDF

Descripción

int **fdf_create** (void)

La función **fdf_create()** crea un documento FDF nuevo. Esta función se necesita si se desea rellenar los campos de entrada en un documento PDF.

Ejemplo 1. Rellenando un documento PDF

```
<?php
$outfdf = fdf_create();
fdf_set_value($outfdf, "volumen", $volume, 0);

fdf_set_file($outfdf, "http:/testfdf/resultlabel.pdf");
fdf_save($outfdf, "outtest.fdf");
fdf_close($outfdf);
Header("Content-type: application/vnd.fdf");
$fp = fopen("outtest.fdf", "r");
fpassthru($fp);
unlink("outtest.fdf");
?>
```

Vea también **fdf_close()**, **fdf_save()**, **fdf_open()**.

fdf_enum_values

(PHP 4 >= 4.3.0, PHP 5)

fdf_enum_values -- Call a user defined function for each document value

Description

bool **fdf_enum_values** (resource fdfdoc, callback function [, mixed userdata])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

fdf_errno

(PHP 4 >= 4.3.0, PHP 5)

fdf_errno -- Return error code for last fdf operation

Description

int **fdf_errno** (void)

fdf_errno() returns the error code set by the last FDF function call. This is zero for a successfull operation or a non-zero error code on failure. A textual description may be obtained using the **fdf_error()** function.

See also **fdf_error()**.

fdf_error

(PHP 4 >= 4.3.0, PHP 5)

fdf_error -- Return error description for fdf error code

Description

string **fdf_error** ([int error_code])

fdf_error() returns a textual description for the fdf error code given in *error_code*. The function uses the internal error code set by the last operation if no *error_code* is given, so *fdf_error()* is a convenient shortcut for *fdf_error(fdf_errno())*.

See also **fdf_errno()**.

fdf_get_ap

(PHP 4 >= 4.3.0, PHP 5)

fdf_get_ap -- Get the appearance of a field

Description

bool **fdf_get_ap** (resource fdf_document, string field, int face, string filename)

The **fdf_get_ap()** function gets the appearance of a *field* (i.e. the value of the /AP key) and stores it in a file. The possible values of *face* are **`FDFNormalAP`**, **`FDFRolloverAP`** and **`FDFDownAP`**. The appearance is stored in *filename*.

fdf_get_attachment

(PHP 4 >= 4.3.0, PHP 5)

fdf_get_attachment -- Extracts uploaded file embedded in the FDF

Description

array **fdf_get_attachment** (resource fdf_document, string fieldname, string savepath)

Extracts a file uploaded by means of the "file selection" field *fieldname* and stores it under *savepath*. *savepath* may be the name of a plain file or an existing directory in which the file is to be created under its original name. Any existing file under the same name will be overwritten.

> **Nota:** There seems to be no other way to find out the original filename but to store the file using a directory as *savepath* and check for the basename it was stored under.

The returned array contains the following fields:

- *path* - path were the file got stored

  size - size of the stored file in bytes

  type - mimetype if given in the FDF

Ejemplo 1. Storing an uploaded file

```
<?php
  $fdf = fdf_open_string($HTTP_FDF_DATA);
  $data = fdf_get_attachment($fdf, "filename", "/tmpdir");
  echo "The uploaded file is stored in $data[path]";
?>
```

fdf_get_encoding

(PHP 4 >= 4.3.0, PHP 5)

fdf_get_encoding -- Get the value of the /Encoding key

Description

string **fdf_get_encoding** (resource fdf_document)

The **fdf_get_encoding()** returns the value of the /Encoding key. An empty string is returned if the default PDFDocEncoding/Unicode scheme is used.

See also **fdf_set_encoding()**.

fdf_get_file

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

fdf_get_file -- Obtener el valor de la clave /F

Descripción

string **fdf_get_file** (int fdf_document)

La función **fdf_set_file()** devuelve el valor de la clave /F.

Vea también **fdf_set_file()**.

fdf_get_flags

(PHP 4 >= 4.3.0, PHP 5)

fdf_get_flags -- Gets the flags of a field

Description

int **fdf_get_flags** (resource fdfdoc, string fieldname, int whichflags)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

fdf_get_opt

(PHP 4 >= 4.3.0, PHP 5)

fdf_get_opt -- Gets a value from the opt array of a field

Description

mixed **fdf_get_opt** (resource fdfdof, string fieldname [, int element])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

fdf_get_status

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

fdf_get_status -- Obtener el valor de la clave /STATUS

Descripción

string **fdf_get_status** (int fdf_document)

La función **fdf_get_status()** devuelve el valor de la clave /STATUS.

Vea también **fdf_set_status()**.

fdf_get_value

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

fdf_get_value -- Obtener el valor de un campo

Descripción

string **fdf_get_value** (int fdf_document, string fieldname)

La función **fdf_get_value()** devuelve el valor de un campo.

Vea también **fdf_set_value()**.

fdf_get_version

(PHP 4 >= 4.3.0, PHP 5)

fdf_get_version -- Gets version number for FDF API or file

Description

string **fdf_get_version** ([resource fdf_document])

This function will return the fdf version for the given *fdf_document*, or the toolkit API version number if no parameter is given.

For the current FDF toolkit 5.0 the API version number is '5.0' and the document version number is either '1.2', '1.3' or '1.4'.

See also **fdf_set_version()**.

fdf_header

(PHP 4 >= 4.3.0, PHP 5)

fdf_header -- Sets FDF-specific output headers

Description

bool **fdf_header** (void)

This is a convenience function to set appropriate HTTP headers for FDF output. It sets the *Content-type:* to *application/vnd.fdf*.

fdf_next_field_name

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

fdf_next_field_name -- Obtener el nombre del siguiente campo

Descripción

string **fdf_next_field_name** (int fdf_document, string fieldname)

La función **fdf_next_field_name()** devuelve el nombre del campo tras el campo *fieldname* o el nombre del primer campo si el segundo parámetro es **NULL**.

Vea también **fdf_set_field()**, **fdf_get_field()**.

fdf_open_string

(PHP 4 >= 4.3.0, PHP 5)

fdf_open_string -- Read a FDF document from a string

Description

resource **fdf_open_string** (string fdf_data)

The **fdf_open_string()** function reads form data from a string. *fdf_data* must contain the data as returned from a PDF form or created using **fdf_create()** and **fdf_save_string()**.

You can **fdf_open_string()** together with *$HTTP_FDF_DATA* to process fdf form input from a remote client.

Ejemplo 1. Accessing the form data

```
<?php
$fdf = fdf_open_string($HTTP_FDF_DATA);
/* ... */
fdf_close($fdf);
?>
```

See also **fdf_open()**, **fdf_close()**, **fdf_create()** and **fdf_save_string()**.

fdf_open

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

fdf_open -- Abrir un documento FDF

Descripción

int **fdf_open** (string filename)

La función **fdf_open()** abre un archivo con datos de formulario. Este archivo debe contener los

datos tal y como se devuelven en un formulario PDF. Actualmente dicho archivo debe crearse "manualmente" usando la función **fopen()** y escribiendo en éste el contenido de HTTP_FDF_DATA usando **fwrite()**. No existe un mecanismo similar al de los formularios HTML donde se crea una variable para cada campo de entrada.

Ejemplo 1. Accediendo a los datos del formulario

```
<?php
// Guarda los datos FDF en un archivo temporal
$fdffp = fopen("test.fdf", "w");
fwrite($fdffp, $HTTP_FDF_DATA, strlen($HTTP_FDF_DATA));
fclose($fdffp);

// Abre archivo temporal y evalúa los datos
$fdf = fdf_open("test.fdf");
...
fdf_close($fdf);
?>
```

Vea también **fdf_close()**.

fdf_remove_item

(PHP 4 >= 4.3.0, PHP 5)

fdf_remove_item -- Sets target frame for form

Description

bool **fdf_remove_item** (resource fdfdoc, string fieldname, int item)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

fdf_save_string

(PHP 4 >= 4.3.0, PHP 5)

fdf_save_string -- Returns the FDF document as a string

Description

string **fdf_save_string** (resource fdf_document)

The **fdf_save_string()** function returns the FDF document as a string.

Ejemplo 1. Retrieving FDF as a string

```
<?php
$fdf = fdf_create();
fdf_set_value($fdf, "foo", "bar");
$str = fdf_save_string($fdf);
fdf_close($fdf);
echo $str;
?>
```

will output something like

```
%FDF-1.2
%ÃƒÂ¢ÃƒÂ£ÃƒÂ  Ãƒâ€œ
1 0 obj
<<
/FDF << /Fields 2 0 R >>
>>
endobj
2 0 obj
[
<< /T (foo)/V (bar)>>
]
endobj
trailer
<<
/Root 1 0 R

>>
%%EOF
```

See also **fdf_save()**, **fdf_open_string()**, **fdf_create()** and **fdf_close()**.

fdf_save

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

fdf_save -- Guardar un documeto FDF

Descripción

int **fdf_save** (string filename)

La función **fdf_save()** guarda un documento FDF. El kit de FDF proporciona una forma de volcar el documento a stdout si el parámetro *filename* es '.'. Esto no funciona si el PHP se utiliza como un módulo del apache. En tal caso se deberá escribir a un fichero y utilizar p. ej. **fpassthru()** para visualizarlo.

Vea también **fdf_close()** y el ejemplo para **fdf_create()**.

fdf_set_ap

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

fdf_set_ap -- Ajusta la apariencia de un campo

Descripción

void **fdf_set_ap** (int fdf_document, string field_name, int face, string filename, int page_number)

La función **fdf_set_ap()** ajusta la apariencia de un campo (p. ej. el valor de la clave /AP). Los valores posibles de *face* son 1=FDFNormalAP, 2=FDFRolloverAP, 3=FDFDownAP.

fdf_set_encoding

(PHP 4 >= 4.1.0, PHP 5)

fdf_set_encoding -- Sets FDF character encoding

Description

bool **fdf_set_encoding** (resource fdf_document, string encoding)

fdf_set_encoding() sets the character encoding in FDF document *fdf_document*. *encoding* should be the valid encoding name. Currently the following values are supported: "*Shift-JIS*", "*UHC*", "*GBK*","*BigFive*". An empty string resets the encoding to the default PDFDocEncoding/Unicode scheme.

fdf_set_file

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

fdf_set_file -- Fijar el valor de la clave /F

Descripción

void **fdf_set_file** (int fdf_document, string filename)

La función **fdf_set_file()** fija el valor de la clave /F. La clave /F es simplemente una referencia a un formulario PDF que se va a rellenar con datos. En un entorno web es un URL (p.ej. http:/testfdf/resultlabel.pdf).

Vea también **fdf_get_file()** y el ejemplo para **fdf_create()**.

fdf_set_flags

(PHP 4 >= 4.0.2, PHP 5)

fdf_set_flags -- Sets a flag of a field

Description

bool **fdf_set_flags** (resource fdf_document, string fieldname, int whichFlags, int newFlags)

The **fdf_set_flags()** sets certain flags of the given field *fieldname*.

See also **fdf_set_opt()**.

fdf_set_javascript_action

(PHP 4 >= 4.0.2, PHP 5)

fdf_set_javascript_action -- Sets an javascript action of a field

Description

bool **fdf_set_javascript_action** (resource fdf_document, string fieldname, int trigger, string script)

fdf_set_javascript_action() sets a javascript action for the given field *fieldname*.

See also **fdf_set_submit_form_action()**.

fdf_set_on_import_javascript

(PHP 4 >= 4.3.0, PHP 5)

fdf_set_on_import_javascript -- Adds javascript code to be executed when Acrobat opens the FDF

Description

bool **fdf_set_on_import_javascript** (resource fdfdoc, string script [, bool before_data_import])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

See also **fdf_add_doc_javascript()**, y **fdf_set_javascript_action()**.

fdf_set_opt

(PHP 4 >= 4.0.2, PHP 5)

fdf_set_opt -- Sets an option of a field

Description

bool **fdf_set_opt** (resource fdf_document, string fieldname, int element, string str1, string str2)

The **fdf_set_opt()** sets options of the given field *fieldname*.

See also **fdf_set_flags()**.

fdf_set_status

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

fdf_set_status -- Fija el valor de la clave /STATUS

Descripción

void **fdf_set_status** (int fdf_document, string status)

La función **fdf_set_status()** fija el valor de la clave /STATUS.

Vea también **fdf_get_status()**.

fdf_set_submit_form_action

(PHP 4 >= 4.0.2, PHP 5)

fdf_set_submit_form_action -- Sets a submit form action of a field

Description

bool **fdf_set_submit_form_action** (resource fdf_document, string fieldname, int trigger, string script, int flags)

The **fdf_set_submit_form_action()** sets a submit form action for the given field *fieldname*.

See also **fdf_set_javascript_action()**.

fdf_set_target_frame

(PHP 4 >= 4.3.0, PHP 5)

fdf_set_target_frame -- Set target frame for form display

Description

bool **fdf_set_target_frame** (resource fdf_document, string frame_name)

Sets the target frame to display a result PDF defined with **fdf_save_file()** in.

See also **fdf_save_file()**.

fdf_set_value

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

fdf_set_value -- Fijar el valor de un campo

Descripción

void **fdf_set_value** (int fdf_document, string fieldname, string value, int isName)

La función **fdf_set_value()** fija el valor de un campo. El parámetro final determina si el valor del campo se deberá convertir a un Nombre PDF (*isName* = 1) o convertir en una Cadena PDF (*isName* = 0).

Vea también **fdf_get_value()**.

fdf_set_version

(PHP 4 >= 4.3.0, PHP 5)

fdf_set_version -- Sets version number for a FDF file

Description

string **fdf_set_version** (resource fdf_document, string version)

This function will set the fdf *version* for the given *fdf_document*. Some features supported by this extension are only available in newer fdf versions. For the current FDF toolkit 5.0 *version* may be either '1.2', '1.3' or '1.4'.

See also **fdf_get_version()**.

XXXVI. Funciones filePro

Estas funciones permiten acceso en modo de solo-lectura a datos guardados en bases de datos filePro.

filePro es una marca registrada de fP Technologies, Inc. Mas informacion sobre filePro puede encontrarse en http://www.fptech.com/.

Tabla de contenidos

filepro_fieldcount

(PHP 3, PHP 4 , PHP 5)

filepro_fieldcount -- encuentra cuantos campos existen en una base de datos filePro

Descripcion

int **filepro_fieldcount** (void)

Devuelve el numero de campos (columnas) existentes en la base de datos filePro abierta.

Ver tambien **filepro()**.

filepro_fieldname

(PHP 3, PHP 4 , PHP 5)

filepro_fieldname -- obtiene el nombre de un campo

Descripcion

string **filepro_fieldname** (int field_number)

Devuelve el nombre del campo correspondiente a *field_number*.

filepro_fieldtype

(PHP 3, PHP 4 , PHP 5)

filepro_fieldtype -- obtiene el tipo de campo

Descripcion

string **filepro_fieldtype** (int field_number)

Devuelve el tipo de campo del campo correspondiente a *field_number*.

filepro_fieldwidth

(PHP 3, PHP 4 , PHP 5)

filepro_fieldwidth -- obtiene la anchura de un campo

Descripcion

int **filepro_fieldwidth** (int field_number)

Devuelve la anchura de el campo correspondiente a *field_number*.

filepro_retrieve

(PHP 3, PHP 4 , PHP 5)

filepro_retrieve -- extrae informacion de una base de datos filePro

Descripcion

string **filepro_retrieve** (int row_number, int field_number)

Devuelve la informacion de la base de datos contenida en la localizacion especificada.

filepro_rowcount

(PHP 3, PHP 4 , PHP 5)

filepro_rowcount -- encuentra cuantas filas existen en una base de datos filePro

Descripcion

int **filepro_rowcount** (void)

Devuelve el numero de filas (entradas) existentes en la base de datos filePro abierta.

Ver tambien **filepro()**.

filepro

(PHP 3, PHP 4 , PHP 5)

filepro -- lee y verifica el fichero de mapeo

Descripcion

bool **filepro** (string directory)

Lee y verifica el fichero de mapeo, guardando la relacion de campos e informacion.

Ningun bloqueo es realizado, por ello, no se deberia modificar la base de datos filePro cuando puede ser abierta con PHP.

XXXVII. Funciones del Sistema de Archivos

Introducción

Requirimientos

No se requieren bibliotecas externas para compilar esta extensión, pero si desea que PHP ofrezca soporte para LFS (archivos grandes) en Linux, entonces necesita tener una versión reciente de glibc y necesita compilar PHP con las siguientes banderas del compilador: *-D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64*.

Instalación

No se necesita ninguna instalación para usar estas funciones, son parte del núcleo de PHP.

Configuración en tiempo de ejecución

El comportamiento de estas funciones está afectado por los valores definidos en `php.ini`.

Tabla 1. Opciones de Configuración del Sistema de Archivos y Secuencias

Nombre	Predeterminado	Modificable
allow_url_fopen	"1"	PHP_INI_SYSTEM
user_agent	NULL	PHP_INI_ALL
default_socket_timeout	"60"	PHP_INI_ALL
from	NULL	??
auto_detect_line_endings	"Off"	PHP_INI_ALL

A continuación se presenta una corta explicación de las directivas de configuración.

allow_url_fopen **boolean**

Esta opción habilita las envolturas fopen tipo URL que le permiten acceder a objetos URL como archivos. Existen envolturas predeterminadas para el acceso de archivos remotos usando los protocolos ftp o http, algunas extensiones como zlib pueden registrar envolturas adicionales.

Nota: Este parámetro puede ser definido únicamente en php.ini debido a razones de seguridad.

Nota: Esta opción fue introducida inmediatamente después del lanzamiento de la versión 4.0.3. Para versiones anteriores a, e incluyendo la 4.0.3, solo puede deshabilitar esta característica en tiempo de compilación usando el parámetro de configuración *--disable-url-fopen-wrapper*.

Aviso
En versiones de windows anteriores a PHP 4.3.0, las siguientes funciones no soportan el acceso de archivos remotos: **include()**, **include_once()**, **require()**, **require_once()** y las funciones imagecreatefromXXX en la extensión Referencia LII, *Funciones para imágenes*.

user_agent **string**

Definir el agente de usuario que PHP envía.

default_socket_timeout **integer**

Tiempo de espera predeterminado (en segundos) para secuencias basadas en sockets.

Nota: Esta opción de configuración fue introducida en PHP 4.3.0

from="joe@example.com" **string**

Definir la contraseña de ftp anónimo (su dirección de correo electrónico).

auto_detect_line_endings **boolean**

Cuando está habilitada esta opción, PHP examina los datos leídos por **fgets()** y **file()** para ver si está usando convenciones de final de líneo tipo Unix, MS-Dos o Macintosh.

Esto le permite a PHP interoperar con sistemas Macintosh, pero tiene un valor predeterminado de Off, ya que hay un impacto ligero de rendimiento cuando se detectan las convenciones EOL para la primera línea, y también porque las personas que usen retornos de carro como separador de elementos bajo sistemas Unix percibirían un comportamiento que no es compatible con versiones anteriores.

Nota: Esta opción de configuración fue introducida en PHP 4.3.0

Tipos de recursos

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

GLOB_BRACE (integer)

GLOB_ONLYDIR (integer)

GLOB_MARK (integer)

GLOB_NOSORT (integer)

GLOB_NOCHECK (integer)

GLOB_NOESCAPE (integer)

PATHINFO_DIRNAME (integer)

PATHINFO_BASENAME (integer)

PATHINFO_EXTENSION (integer)

FILE_USE_INCLUDE_PATH (integer)

FILE_APPEND (integer)

FILE_IGNORE_NEW_LINES (integer)

FILE_SKIP_EMPTY_LINES (integer)

Ver también

Para funciones relacionadas, vea también las secciones Directorio y Ejecución de Programas.

Para una lista de las varias envolturas de URL que pueden ser usadas como archivos remotos, y su explicación, vea también Apéndice L.

Tabla de contenidos

is_executable -- Indica si el archivo es ejecutable
is_file -- Indica si el archivo es un archivo regular
is_link -- Indica si el archivo es un enlace simbólico
is_readable -- Indica si es posible leer el archivo
is_uploaded_file -- Indica si un archivo fue cargado a través de HTTP POST
is_writable -- Indica si el nombre de archivo es escribible
is_writeable -- Alias de **is_writable()**
link -- Crea un enlace fuerte
linkinfo -- Consigue información sobre un enlace
lstat -- Entrega información sobre un archivo o enlace simbólico
mkdir -- Crea un directorio
move_uploaded_file -- Mueve un archivo cargado a una nueva ubicación
parse_ini_file -- Procesar un archivo de configuración
pathinfo -- Devuelve información sobre la ruta de un archivo
pclose -- Cierra un apuntador de archivo de proceso
popen -- Abre un apuntador de archivo de proceso
readfile -- Imprime un archivo
readlink -- Devuelve el objetivo de un enlace simbólico
realpath -- Devuelve el nombre de ruta absoluto simplificado
rename -- Renombra un archivo o directorio
rewind -- Retroceder la posición de un apuntador de archivo
rmdir -- Elimina un directorio
set_file_buffer -- Alias de **stream_set_write_buffer()**
stat -- Entrega información sobre un archivo
symlink -- Crea un enlace simbólico
tempnam -- Crear un archivo con un nombre único
tmpfile -- Crea un archivo temporal
touch -- Establece la hora de acceso y modificación de un archivo
umask -- Cambia la umask actual
unlink -- Elimina un archivo

basename

(PHP 3, PHP 4 , PHP 5)

basename -- Devuelve la parte del path correspondiente al nombre del fichero

Descripción

string **basename** (string path)

Dada una cadena (string) que contiene el path de un fichero, esta función devuelve el nombre base del fichero.

En Windows, tanto la barra (/) como la barra inversa (\) pueden usarse como caracter separador en el path. En otros entornos, se usa la barra directa (/).

Ejemplo 1. Ejemplo de basename()

```
$path = "/home/httpd/html/index.php3";
$file = basename($path); // $file toma el valor "index.php3"
```

Ver también: **dirname()**

chgrp

(PHP 3, PHP 4 , PHP 5)

chgrp -- Cambia el grupo de un fichero

Descripción

int **chgrp** (string filename, mixed group)

Trata de cambiar el grupo al que pertenece el fichero filename al grupo group. Sólo el superusuario puede cambiar el grupo de un fichero arbitrariamente; otros usuarios pueden cambiar el grupo del fichero a cualquier grupo del cual el usuario sea miembro.

Devuelve **TRUE** en caso de éxito; en otro caso devuelve **FALSE**.

En Windows, no hace nada y devuelve **TRUE**.

Ver también **chown()** y **chmod()**.

chmod

(PHP 3, PHP 4 , PHP 5)

chmod -- Cambia permisos de un fichero

Descripción

int **chmod** (string filename, int mode)

Trata de cambiar los permisos del fichero especificado por *filename* a los permisos dados por *mode*.

Cuidado que *mode* no es asumido de forma automática como un valor octal. Para asegurar que se hace la operación esperada necesitas anteponer un cero (0) como prefijo del parámetro *mode*:

```
chmod( "/somedir/somefile", 755 );   // decimal; probablemente incorrecto
chmod( "/somedir/somefile", 0755 );  // octal; valor correcto de mode
```

Devuelve **TRUE** en caso de éxito y **FALSE** en otro caso.

Ver también **chown()** y **chgrp()**.

chown

(PHP 3, PHP 4 , PHP 5)

chown -- Cambia el propietario de un fichero

Descripción

int **chown** (string filename, mixed user)

Trata de cambiar el propietario del fichero filename al usuario user. Sólo el superusuario puede cambiar el propietario de un fichero.

Devuelve **TRUE** en caso de éxito; en otro caso devuelve **FALSE**.

> **Nota:** En Windows, no hace nada y devuelve **TRUE**.

Ver también **chown()** y **chmod()**.

clearstatcache

(PHP 3, PHP 4 , PHP 5)

clearstatcache -- Limpia la cache de estado de un fichero

Descripción

void **clearstatcache** (void)

Invocar la llamada del sistema stat o lstat es bastante costoso en la mayoría de los sistemas. Por lo tanto, el resultado de la última llamada a cualquiera de las funciones de estado (listadas abajo) es guardado para usarlo en la próxima llamada de este tipo empleando el mismo nombre de fichero. Si deseas forzar un nuevo chequeo del estado del fichero, por ejemplo si el fichero está siendo chequeado muchas veces y puede cambiar o desaparecer, usa esta función para borrar los resultados almacenados en memoria de la última llamada.

Este valor sólo es cacheado durante el tiempo de vida de una petición simple.

Entre las funciones afectadas se incluyen **stat()**, **lstat()**, **file_exists()**, **is_writeable()**, **is_readable()**, **is_executable()**, **is_file()**, **is_dir()**, **is_link()**, **filectime()**, **fileatime()**, **filemtime()**, **fileinode()**, **filegroup()**, **fileowner()**, **filesize()**, **filetype()**, y **fileperms()**.

copy

(PHP 3, PHP 4 , PHP 5)

copy -- Copia un fichero

Descripción

int **copy** (string source, string dest)

Hace una copia de un fichero. Devuelve **TRUE** si la copia tiene éxito, y **FALSE** en otro caso.

Ejemplo 1. Ejemplo de copy()

```
if (!copy($file, $file.'.bak')) {
    print("failed to copy $file...<br>\n");
}
```

Ver también: **rename()**.

delete

(no version information, might be only in CVS)

delete -- Una entrada manual inútil

Descripción

void **delete** (string file)

Esto es una entrada manual inútil para satisfacer a esas personas que están buscando **unlink()** o **unset()** en el lugar ñocado.

Ver también: **unlink()** para borrar ficheros, **unset()** para borrar variables.

dirname

(PHP 3, PHP 4 , PHP 5)

dirname -- Devuelve la parte del path correspondiente al directorio

Descripción

string **dirname** (string path)

Dada una cadena (string) conteniendo el path a un fichero, esta función devolverá el nombre del directorio.

En Windows, tanto la barra (/) como la barra inversa (\) son usadas como separadores de caracteres. En otros entornos, debe usarse la barra directa (/).

Ejemplo 1. Ejemplo de dirname()

```
$path = "/etc/passwd";
$file = dirname($path); // $file toma el valor "/etc"
```

Ver también: **basename()**

disk_free_space

(PHP 4 >= 4.1.0, PHP 5)

disk_free_space -- Devuelve el espacio disponible en el directorio

Descripción

float **disk_free_space** (string directorio)

Dada una cadena que contiene un directorio, esta función devolverá el número de bytes disponibles

en el sistema de archivos o partición de disco correspondiente.

Ejemplo 1. Ejemplo de disk_free_space()

```
<?php
// $df contiene el numero de bytes disponibles en "/"
$df = disk_free_space("/");

// En Windows:
disk_free_space("C:");
disk_free_space("D:");
?>
```

> **Nota:** Esta funcion no funcionara con ficheros remotos ya que el fichero a examinar tiene que estar disponible desde el sistema de ficheros del servidor.

Vea también **disk_total_space()**

disk_total_space

(PHP 4 >= 4.1.0, PHP 5)

disk_total_space -- Devuelve el tamaño total de un directorio

Descripción

float **disk_total_space** (string directorio)

Dada una cadena que contiene un directorio, esta función devolverá el número total de bytes en el sistema de archivos o partición de disco correspondiente.

Ejemplo 1. Ejemplo de disk_total_space()

```
<?php
// $df contiene el numero total de bytes disponible en "/"
$df = disk_total_space("/");

// En Windows:
disk_total_space("C:");
disk_total_space("D:");
?>
```

> **Nota:** Esta funcion no funcionara con ficheros remotos ya que el fichero a examinar tiene que estar disponible desde el sistema de ficheros del servidor.

Vea también **disk_free_space()**

diskfreespace

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

diskfreespace -- Devuelve el espacio disponible en un directorio

Descripción

float **diskfreespace** (string directory)

Dada una cadena (string) conteniendo el nombre de un directorio, esta función devolverá el número de bytes disponibles en el disco correspondiente.

Ejemplo 1. Ejemplo de diskfreespace()

```
$df = diskfreespace("/"); // $df contiene el numero de bytes
                          // disponibles en "/"
```

fclose

(PHP 3, PHP 4 , PHP 5)

fclose -- Cierra el apuntador a un fichero abierto

Description

int **fclose** (int fp)

Se cierra el fichero apuntado por fp.

Devuelve **TRUE** en caso de éxito y **FALSE** en caso de fallo.

El apuntador al fichero debe ser válido y debe apuntarse a un fichero abierto con éxito con **fopen()** o con **fsockopen()**.

feof

(PHP 3, PHP 4 , PHP 5)

feof -- Verifica si el apuntador a un fichero está al final del fichero (end-of-file)

Descripción

int **feof** (int fp)

Devuelve **TRUE** si el apuntador del fichero está en EOF o si ocurre un error; en otro caso devuelve **FALSE**.

The file pointer must be valid, and must point to a file El apuntador al fichero debe ser válido, y debe apuntar a un fichero abierto con éxito por **fopen()**, **popen()**, o **fsockopen()**.

fflush

(PHP 4 >= 4.0.1, PHP 5)

fflush -- Vacía la salida hacia un archivo

Descripción

bool **fflush** (resource gestor)

Esta función obliga a que se produzca la escritura de la salida acumulada en búfer al recurso apuntado por el gestor de archivo *gestor*. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

El puntero de fichero debe de ser valido y debe de apuntar a un fichero abierto con exito por **fopen()** o **fsockopen()**.

fgetc

(PHP 3, PHP 4 , PHP 5)

fgetc -- Obtiene un caracter del fichero apuntado

Descripción

string **fgetc** (int fp)

Devuelve una cadena (string) conteniendo un simple caracter leido del fichero apuntado por fp. Devuelve **FALSE** para EOF (como hace **feof()**).

El apuntador al fichero debe ser valido, y debe apuntar a un fichero abierto con éxito por **fopen()**, **popen()**, o **fsockopen()**.

Ver también **fread()**, **fopen()**, **popen()**, **fsockopen()**, y **fgets()**.

fgetcsv

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

fgetcsv -- Obtiene una línea del fichero apuntado y extrae los campos CSV

Descripción

array **fgetcsv** (int fp, int length [, string delimiter])

Parecida a fgets() excepto que fgetcsv() parsea la línea que lee buscando campos en formato CSV y devuelve un array conteniendo los campos leidos. El delimitador de campo es una coma, a menos que se especifique otro delimitador con el tercer parámetro opcional.

fp debe ser un apuntador válido a un fichero abierto con éxito por **fopen()**, **popen()**, o **fsockopen()**

la longitud debe ser mayor que la línea más larga que pueda encontrarse en le fichero CSV (permitiendo arrastrar caracteres de fin de línea)

fgetcsv() devuelve **FALSE** en caso de error, incluyendo el fin de fichero.

NOTA: Una línea en blanco en un fichero CSV se devuelve como un array que contiene un único campo nulo, y esto no será tratado como un error.

Ejemplo 1. Ejemplo de fgetcsv() - Leer e imprimir el contenido completo de un fichero CSV

```
<?php
$row = 1;
$fp = fopen ("test.csv","r");
while ($data = fgetcsv ($fp, 1000, ",")) {
    $num = count ($data);
    print "<p> $num fields in line $row: <br />";
    $row++;
    for ($c=0; $c<$num; $c++) {
        print $data[$c] . "<br />";
    }
}
fclose ($fp);
?>
```

fgets

(PHP 3, PHP 4 , PHP 5)

fgets -- Obtiene una línea desde el apuntador de archivo

Descripción

string **fgets** (resource gestor [, int longitud])

Devuelve una cadena de hasta *longitud* - 1 bytes leídos desde el archivo apuntado por *gestor*. La lectura termina cuando se han leído *longitud* - 1 bytes, se alcanza un salto de línea (el cual se incluye en el valor devuelto), o en EOF (lo que ocurra primero). Si no se especifica una longitud, el valor predeterminado es de 1k, o 1024 bytes.

Si ocurre un error, devuelve **`FALSE`**.

Errores comunes:

Aquellos acostumbrados a la semántice de **fgets()** en 'C', debe notar la diferencia en el modo en que *EOF* es devuelto.

El puntero de fichero debe de ser valido y debe de apuntar a un fichero abierto con exito por **fopen()** o **fsockopen()**.

A continuación se presenta un ejemplo simple:
Ejemplo 1. Lectura de un archivo línea a línea

```
<?php
$gestor = fopen("/tmp/archivo_entrada.txt", "r");
while (!feof($gestor)) {
    $bufer = fgets($gestor, 4096);
    echo $bufer;
}
fclose($gestor);
?>
```

> **Nota:** El parámetro *longitud* se hizo opcional en PHP 4.2.0, si se omite, se asume 1024 como la longitud de línea. A partir de PHP 4.3, al omitir *longitud*, la lectura de la secuencia continuará hasta que se alcance el final de la línea. Si la mayoría de líneas en el archivo superan los 8KB, es más eficiente en términos de recursos espicificar la longitud máxima de línea en su script.
>
> **Nota:** Esta función es segura con material binario desde PHP 4.3. Las versiones

anteriores no contaban con ésta característica.

Nota: Si sufre problemas con *PHP* no reconociendo los finales de línea cuando lee archivos creados en un Macintosh (o leyendo archivos sobre uno), puede probar activando la opción de configuración auto_detect_line_endings.

Vea también **fread()**, **fgetc()**, **stream_get_line()**, **fopen()**, **popen()**, **fsockopen()**, y **stream_set_timeout()**.

fgetss

(PHP 3, PHP 4 , PHP 5)

fgetss -- Obtiene una lína desde el apuntador de archivo y elimina las etiquetas HTML

Descripción

string **fgetss** (resource gestor [, int longitud [, string etiquetas_permisibles]])

Idéntica a **fgets()**, excepto que fgetss intenta eliminar cualquier etiqueta HTML y PHP del texto que lee.

Puede usar el opcional tercer parámetro para especificar las etiquetas que no desea remover.

Nota: *etiquetas_permisibles* se agregó en PHP 3.0.13, PHP 4.0.0.

El parámetro *longitud* es opcional desde PHP 5.

Nota: Si sufre problemas con *PHP* no reconociendo los finales de línea cuando lee archivos creados en un Macintosh (o leyendo archivos sobre uno), puede probar activando la opción de configuración auto_detect_line_endings.

Vea también **fgets()**, **fopen()**, **fsockopen()**, **popen()**, y **strip_tags()**.

file_exists

(PHP 3, PHP 4 , PHP 5)

file_exists -- Verifica si un archivo o directorio existe

Descripción

bool **file_exists** (string nombre_archivo)

Devuelve **TRUE** si el archivo o directorio especificado por *nombre_archivo* existe; o **FALSE** de lo contrario.

En windows, use `//nombre_computadora/recurso/nombre_archivo` o `\\nombre_computadora\recurso\nombre_archivo` para revisar archivos en recursos compartidos de red.

Ejemplo 1. Probar si un archivo existe

```
<?php
$nombre_archivo = '/ruta/hacia/foo.txt';

if (file_exists($nombre_archivo)) {
    echo "El archivo $nombre_archivo existe";
} else {
    echo "El archivo $nombre_archivo no existe";
}
?>
```

> **Nota:** Los resultados de esta función son guardados. Consultar **clearstatcache()** para más detalles.
>
> **Sugerencia:** A partir de PHP 5.0.0, esta funcion tambien puede usarse con *algunas* URL como nombre de fichero. Consultar Apéndice L, para obtener una lista con soporte para la funcionalidad **stat()**.

Vea también **is_readable()**, **is_writable()**, **is_file()** y **file()**.

file_get_contents

(PHP 4 >= 4.3.0, PHP 5)

file_get_contents -- Lee un archivo entero en una cadena

Descripción

string **file_get_contents** (string nombre_archivo [, bool usar_ruta_inclusion [, resource contexto]])

Función idéntica a **file()**, con la excepción de que **file_get_contents()** devuelve el archivo en una cadena. En caso de fallo, **file_get_contents()** devolverá **FALSE**.

file_get_contents() es el modo preferido para leer los contenidos de un archivo en una cadena. Esta función usa las técnicas de asignación de memoria que soporte su SO para incrementar su rendimiento.

Ejemplo 1. Uso de file_get_contents() con una URI

Si está abriendo una URI con caracteres especiales, como espacios, necesita codificar la URI con **urlencode()**.

```
<?php
$url = 'http://example.com/con caracteres chistosos';
list($protocolo, $uri) = split('//', $url);
$html = file_get_contents($protocolo . '//' . urlencode($uri));
?>
```

> **Nota:** Esta función es segura binariamente.
>
> **Sugerencia:** Puede usar una URL como nombre de archivo con esta función si los fopen wrappers han sido activados. Consulte **fopen()** para más detalles sobre cómo especificar el nombre de fichero y Apéndice L una lista de protocolos URL soportados
>
> **Nota:** Soporte de contexto fue introducido con PHP.5.0.0.

Aviso

Cuando se usa SSL, Microsoft IIS violara el protocolo, cerrando la conexion sin mandar un indicador close_notify. PHP avisara de esto con este mensaje "SSL: Fatal Protocol Error", cuando llegue al final de los datos. Una solucion a este problema es bajar el nivel de aviso de errores del sistema para que no incluya advertencias. PHP 4.3.7 y versiones posteriores detectan servidores IIS con este problema y suprime la advertencia. Si usais la funcion **fsockopen()** para crear un socket ssl://, tendreis que suprimir la advertencia explicitamente.

Vea también **fgets()**, **file()**, **fread()**, **include()**, y **readfile()**.

file_put_contents

(PHP 5)

file_put_contents -- Escribir una cadena sobre un archivo

Descripción

int **file_put_contents** (string nombre_archivo, string datos [, int banderas [, resource contexto]])

Idéntico a llamar **fopen()**, **fwrite()**, y **fclose()** sucesivamente. La función devuelve la cantidad de bytes que fueron escritos en el archivo.

banderas puede recibir **`FILE_USE_INCLUDE_PATH`** o **`FILE_APPEND`**, sin embargo, la opción **`FILE_USE_INCLUDE_PATH`** debe ser usada con precaución.

> **Nota:** Esta función es segura binariamente.
>
> **Sugerencia:** Puede usar una URL como nombre de archivo con esta función si los fopen wrappers han sido activados. Consulte **fopen()** para más detalles sobre cómo especificar el nombre de fichero y Apéndice L una lista de protocolos URL soportados

Vea también **fopen()**, **fwrite()**, **fclose()**, y **file_get_contents()**.

file

(PHP 3, PHP 4 , PHP 5)

file -- Lee un archivo entero hacia una matriz

Descripción

array **file** (string nombre_archivo [, int usar_ruta_inclusion [, resource contexto]])

Función idéntica a **readfile()**, excepto que **file()** devuelve el archivo en una matriz. Cada elemento de la matriz corresponde a una línea en el archivo, con el salto de línea aun incluido. Si ocurre un fallo, **file()** devuelve **`FALSE`**.

Es posible usar el parámetro opcional *usar_ruta_inclusion*, y definirlo como "1", si desea buscar por el archivo en include_path, también.

```
<?php
// Obtiene un archivo en una matriz. En este ejemplo usaremos HTTP
// para obtener el codigo fuente HTML de una URL.

$lineas = file('http://www.example.com/');

// Recorrer nuestra matriz, mostrar el codigo HTML como codigo fuente
// HTML, y los numeros de linea tambien.
foreach ($lineas as $linea_num => $linea) {
    echo "L&iacute;nea #<b>{$linea_num}</b> : " . htmlspecialchars($linea) . "<br />\n"
}

// Otro ejemplo, obtengames una pagina web como una cadena. Vea
// tambien file_get_contents().
$html = implode('', file('http://www.example.com/'));
?>
```

Sugerencia: Puede usar una URL como nombre de archivo con esta función si los fopen wrappers han sido activados. Consulte **fopen()** para más detalles sobre cómo especificar el nombre de fichero y Apéndice L una lista de protocolos URL soportados

Nota: Cada línea en la matriz resultante incluye el final de línea, así que aun necesita usar **rtrim()** si no quiere conservar el final de línea.

Nota: Si sufre problemas con *PHP* no reconociendo los finales de línea cuando lee archivos creados en un Macintosh (o leyendo archivos sobre uno), puede probar activando la opción de configuración auto_detect_line_endings.

Nota: A partir de PHP 4.3.0, puede usar **file_get_contents()** para devolver el contenido de un archivo como una cadena.

En PHP 4.3.0 **file()**, se volvió una función segura con material binario.

Nota: Soporte de contexto fue introducido con PHP.5.0.0.

Aviso
Cuando se usa SSL, Microsoft IIS violara el protocolo, cerrando la conexion sin mandar un indicador close_notify. PHP avisara de esto con este mensaje "SSL: Fatal Protocol Error", cuando llegue al final de los datos. Una solucion a este problema es bajar el nivel de aviso de errores del sistema para que no incluya advertencias. PHP 4.3.7 y versiones posteriores detectan servidores IIS con este problema y suprime la advertencia. Si usais la funcion **fsockopen()** para crear un socket ssl://, tendreis que suprimir la advertencia explicitamente.

Vea también **readfile()**, **fopen()**, **fsockopen()**, **popen()**, **file_get_contents()**, y **include()**.

fileatime

(PHP 3, PHP 4 , PHP 5)

fileatime -- Obtiene la hora del último acceso al archivo

Descripción

int **fileatime** (string nombre_archivo)

Devuelve la hora en que se accedió al archivo por última vez, o **FALSE** en caso de un error. La hora es devuelta como una marca de tiempo Unix.

Nota: El valor atime de un archivo debe cambiar cada vez que los bloques de datos de éste sean leídos. Esto puede resultar costoso en términos de rendimiento cuando una aplicación accede regularmente a un número muy grande de archivos o directorios. Algunos sistemas de archivos Unix pueden ser montados deshabilitando los actualizaciones atime para incrementar el rendimiento de tales aplicaciones. Las colas de noticias USENET son un ejemplo común. En tales sistemas de archivos esta función resultará inútil.

> **Nota:** Los resultados de esta función son guardados. Consultar **clearstatcache()** para más detalles.

> **Sugerencia:** A partir de PHP 5.0.0, esta funcion tambien puede usarse con *algunas* URL como nombre de fichero. Consultar Apéndice L, para obtener una lista con soporte para la funcionalidad **stat()**.

Ejemplo 1. Ejemplo de fileatime()

```
<?php

// imprime p.ej. se accedio a un_archivo.txt en: December 29 2002 22:16:23.

$nombre_archivo = 'un_archivo.txt';
if (file_exists($nombre_archivo)) {
    echo "se accedio a $nombre_archivo en: " . date("F d Y H:i:s.", fileatime($nombre_a
}

?>
```

Vea también **filemtime()**, **fileinode()**, y **date()**.

filectime

(PHP 3, PHP 4 , PHP 5)

filectime -- Obtiene la hora de modificación del inode del archivo

Descripción

int **filectime** (string nombre_archivo)

Devuelve la hora en que el archivo fue cambiado por última vez, o **FALSE** en caso de fallo. La hora es devuelta como una marca de tiempo Unix.

Nota: En la mayoría de sistemas de archivos en Unix, un archivo se considera cambiado cuando los datos del inode cambian; es decir, cuando la información de permisos, el dueño, grupo u otros meta-datos son actualizados. Vea también **filemtime()** (que es lo que probablemente quiera usar cuando desea crear pies de página tipo "Modificado por última vez" en páginas web) y **fileatime()**.

Note también que en algunos textos sobre Unix, el valor ctime de un archivo se detalla como la hora de creación del archivo. Esto es falso. No existe una hora de creación para archivos Unix en la mayoría de sistemas de archivos usados en Unix.

> **Nota:** Los resultados de esta función son guardados. Consultar **clearstatcache()** para más detalles.

> **Sugerencia:** A partir de PHP 5.0.0, esta funcion tambien puede usarse con *algunas* URL como nombre de fichero. Consultar Apéndice L, para obtener una lista con soporte

para la funcionalidad **stat()**.

Ejemplo 1. Ejemplo de fileatime()

```
<?php

// imprime p.ej. un_archivo.txt fue modificado en: December 29 2002 22:16:23.

$nombre_archivo = 'un_archivo.txt';
if (file_exists($nombre_archivo)) {
    echo "$nombre_archivo fue modificado en: " . date("F d Y H:i:s.", filectime($nombre
}

?>
```

Vea también **filemtime()**

filegroup

(PHP 3, PHP 4 , PHP 5)

filegroup -- Obtiene el grupo del archivo

Descripción

int **filegroup** (string nombre_archivo)

Devuelve el ID de grupo del archivo, o **FALSE** en caso de fallo. El ID de grupo es devuelto en formato numérico, use **posix_getgrgid()** para convertirlo en un nombre de grupo. En caso de fallo, se devuelve **FALSE** y se genera un error de nivel **E_WARNING**.

> **Nota:** Los resultados de esta función son guardados. Consultar **clearstatcache()** para más detalles.
>
> **Sugerencia:** A partir de PHP 5.0.0, esta funcion tambien puede usarse con *algunas* URL como nombre de fichero. Consultar Apéndice L, para obtener una lista con soporte para la funcionalidad **stat()**.

Vea también **fileowner()**, y safe_mode_gid.

fileinode

(PHP 3, PHP 4 , PHP 5)

fileinode -- Obtiene el inode del archivo

Descripción

int **fileinode** (string nombre_archivo)

Devuelve el número inode del archivo, o **FALSE** en caso de un error.

> **Nota:** Los resultados de esta función son guardados. Consultar **clearstatcache()** para más detalles.

> **Sugerencia:** A partir de PHP 5.0.0, esta funcion tambien puede usarse con *algunas* URL como nombre de fichero. Consultar Apéndice L, para obtener una lista con soporte para la funcionalidad **stat()**.

Vea también **stat()**

filemtime

(PHP 3, PHP 4 , PHP 5)

filemtime -- Obtiene la hora de modificación del archivo

Descripción

int **filemtime** (string nombre_archivo)

Devuelve la hora en que el archivo fue modificado por última vez, o **FALSE** en caso de fallo. La hora es devuelta como una marca de tiempo Unix, la cual es apropiada para la función **date()**.

> **Nota:** Los resultados de esta función son guardados. Consultar **clearstatcache()** para más detalles.

> **Sugerencia:** A partir de PHP 5.0.0, esta funcion tambien puede usarse con *algunas* URL como nombre de fichero. Consultar Apéndice L, para obtener una lista con soporte para la funcionalidad **stat()**.

Esta función devuelve la hora en que los bloques de datos de un archivo recibieron una escritura, es decir, la hora en que el contenido del archivo cambió.

Ejemplo 1. Ejemplo de filemtime()

```
<?php
// imprime, p.ej. un_archivo.txt fue modificado: December 29 2002 22:16:23.

$nombre_archivo = 'un_archivo.txt';
if (file_exists($nombre_archivo)) {
    echo "$nombre_archivo fue modificado: " . date ("F d Y H:i:s.", filemtime($nombre_a
}
?>
```

Vea también **filectime()**, **stat()**, **touch()**, y **getlastmod()**.

fileowner

(PHP 3, PHP 4 , PHP 5)

fileowner -- Obtiene el dueño del archivo

Descripción

int **fileowner** (string nombre_archivo)

Devuelve el ID de usuario del dueño del archivo, o **FALSE** en caso de fallo. El ID de usuario es devuelto en formato numérico, uso **posix_getpwuid()** para convertirlo en un nombre de usuario.

> **Nota:** Los resultados de esta función son guardados. Consultar **clearstatcache()** para más detalles.
>
> **Sugerencia:** A partir de PHP 5.0.0, esta funcion tambien puede usarse con *algunas* URL como nombre de fichero. Consultar Apéndice L, para obtener una lista con soporte para la funcionalidad **stat()**.

Vea también **stat()**

fileperms

(PHP 3, PHP 4 , PHP 5)

fileperms -- Obtiene los permisos del archivo

Descripción

int **fileperms** (string nombre_archivo)

Devuelve los permisos del archivo, o **FALSE** en caso de fallo.

> **Nota:** Los resultados de esta función son guardados. Consultar **clearstatcache()** para más detalles.
>
> **Sugerencia:** A partir de PHP 5.0.0, esta funcion tambien puede usarse con *algunas* URL como nombre de fichero. Consultar Apéndice L, para obtener una lista con soporte para la funcionalidad **stat()**.

Ejemplo 1. Desplegar los permisos como un valor octal

```
<?php
echo substr(sprintf('%o', fileperms('/tmp')), -4);
echo substr(sprintf('%o', fileperms('/etc/passwd')), -4);
?>
```

Esto produciría la salida:

```
1777
0644
```

Ejemplo 2. Mostrar los permisos completos

```
<?php
$perms = fileperms('/etc/passwd');

if (($perms & 0xC000) == 0xC000) {
    // Socket
    $info = 's';
} elseif (($perms & 0xA000) == 0xA000) {
    // Enlace Simbolico
    $info = 'l';
} elseif (($perms & 0x8000) == 0x8000) {
    // Regular
    $info = '-';
} elseif (($perms & 0x6000) == 0x6000) {
    // Bloque especial
    $info = 'b';
} elseif (($perms & 0x4000) == 0x4000) {
    // Directorio
    $info = 'd';
} elseif (($perms & 0x2000) == 0x2000) {
    // Caracter especial
    $info = 'c';
} elseif (($perms & 0x1000) == 0x1000) {
    // Pipe FIFO
    $info = 'p';
} else {
    // Desconocido
    $info = 'u';
}

// Duenyo
$info .= (($perms & 0x0100) ? 'r' : '-');
$info .= (($perms & 0x0080) ? 'w' : '-');
$info .= (($perms & 0x0040) ?
            (($perms & 0x0800) ? 's' : 'x' ) :
            (($perms & 0x0800) ? 'S' : '-'));

// Group
$info .= (($perms & 0x0020) ? 'r' : '-');
$info .= (($perms & 0x0010) ? 'w' : '-');
$info .= (($perms & 0x0008) ?
            (($perms & 0x0400) ? 's' : 'x' ) :
            (($perms & 0x0400) ? 'S' : '-'));

// World
$info .= (($perms & 0x0004) ? 'r' : '-');
$info .= (($perms & 0x0002) ? 'w' : '-');
$info .= (($perms & 0x0001) ?
            (($perms & 0x0200) ? 't' : 'x' ) :
            (($perms & 0x0200) ? 'T' : '-'));

echo $info;
?>
```

Esto produciría la salida:

```
-r--r--r--
```

Vea también **is_readable()**, y **stat()**

filesize

(PHP 3, PHP 4 , PHP 5)

filesize -- Obtiene el tamaño del archivo

Descripción

int **filesize** (string nombre_archivo)

Devuelve el tamaño del archivo en bytes, o **FALSE** en caso de fallo.

> **Nota:** Dado que el tipo entero de PHP tiene signo y muchas plataformas usan enteros de 32 bits, **filesize()** puede devolver resultados inesperados para archivos con un tamaño mayor de 2GB. Para archivos entre 2GB y 4GB de tamaño, esto puede resolverse por lo general usando *sprintf("%u", filesize($archivo))*.
>
> **Nota:** Los resultados de esta función son guardados. Consultar **clearstatcache()** para más detalles.
>
> **Sugerencia:** A partir de PHP 5.0.0, esta funcion tambien puede usarse con *algunas* URL como nombre de fichero. Consultar Apéndice L, para obtener una lista con soporte para la funcionalidad **stat()**.

Ejemplo 1. Ejemplo de filesize()

```
<?php

// imprime, p.ej. un_archivo.txt: 1024 bytes

$nombre_archivo = 'un_archivo.txt';
echo $nombre_archivo . ': ' . filesize($nombre_archivo) . ' bytes';

?>
```

Vea también **file_exists()**

filetype

(PHP 3, PHP 4 , PHP 5)

filetype -- Obtiene el tipo de archivo

Descripción

string **filetype** (string nombre_archivo)

Devuelve el tipo del archivo. Los posibles valores son fifo, char, dir, block, link, file, y unknown.

Devuelve **FALSE** si ocurre un error. **filetype()** producirá siempre un mensaje **E_NOTICE** si el llamado a stat falla o el tipo de archivo es desconocido.

> **Nota:** Los resultados de esta función son guardados. Consultar **clearstatcache()** para más detalles.
>
> **Sugerencia:** A partir de PHP 5.0.0, esta funcion tambien puede usarse con *algunas* URL como nombre de fichero. Consultar Apéndice L, para obtener una lista con soporte para la funcionalidad **stat()**.

Ejemplo 1. Ejemplo de filetype()

```
<?php

echo filetype('/etc/passwd');  // file
echo filetype('/etc/');        // dir

?>
```

Vea también **is_dir()**, **is_file()**, **is_link()**, **file_exists()**, **stat()**, y **mime_content_type()**.

flock

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

flock -- Aviso de bloqueo de archivos portable

Descripción

bool **flock** (resource gestor, int operacion [, int &bloquearia])

PHP soporta una manera portable de bloquear archivos enteros de por medio de avisos (lo que quiere decir que todos los programas que acceden al archivo deben usar el mismo mecanismo de bloqueo, o no funcionará).

> **Nota: flock()** es obligatorio bajo Windows.

flock() opera sobre *gestor*, que debe ser un apuntador a un archivo abierto. *operacion* es uno de los siguientes valores:

- Para adquirir un bloqueo compartido (de lectura), defina *operacion* a **LOCK_SH** (o use 1 antes de PHP 4.0.1).
- Para adquirir un bloqueo exclusizo (de escritura), defina *operacion* a **LOCK_EX** (o use 2 antes de PHP 4.0.1).
- Para liberar un bloqueo (compartido o exclusivo), defina *operacion* a **LOCK_UN** (o use 3 antes de PHP 4.0.1).
- Si no desea que **flock()** bloquee mientras opera, agregue **LOCK_NB** (4 antes de PHP 4.0.1) a *operacion*.

flock() le permite ejecutar un modelo de lectura/escritura simple que puede ser usado en prácticamente cualquier plataforma (incluyendo la mayoría de derivaciones de Unix e incluso Windows). El tercer argumento opcional recibe el valor **TRUE** si el aviso produciría un bloqueo (la condición EWOULDBLOCK)

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplo 1. Ejemplo de flock()

```php
<?php

$aa = fopen("/tmp/bloqueo.txt", "w+");

if (flock($aa, LOCK_EX)) { // realizar un bloqueo exclusivo
    fwrite($aa, "Escribir algo aqui\n");
    flock($aa, LOCK_UN); // liberar el aviso
} else {
    echo "&iexcl;No se pudo bloquear el archivo!";
}

fclose($aa);

?>
```

Nota: Dado que **flock()** requiere un apuntador de archivo, puede que necesite usar un archivo especial de bloqueo para proteger el acceso a un archivo que pretende truncar abriéndolo en modo de escritura (con un argumento "w" o "w+" a **fopen()**).

Aviso
flock() no trabajará sobre NFS y muchos otros sistemas de archivos en red. Revise la documentación de su sistema operativo para más detalles. En algunos sistemas operativos **flock()** se implementa al nivel de proceso. Cuando usa una API de servidor multi-hilos como ISAPI, ¡es posible que no pueda depender en **flock()** para proteger archivos contra otros scripts PHP corriendo en hilos paralelos en la misma instancia de servidor! **flock()** no es soportado en sistemas de archivos anticuados como *FAT* y sus derivados, y por lo tanto siempre devolverá **FALSE** bajo tales entornos (esto es especialmente cierto para usuarios de Windows 98).

fnmatch

(PHP 4 >= 4.3.0, PHP 5)

fnmatch -- Comparar un nombre de archivo contra un patrón

Descripción

bool **fnmatch** (string patron, string cadena [, int banderas])

fnmatch() verifica si la *cadena* pasada coincide con el comodín de intérprete de comandos dado, *patron*.

Esto es especialmente útil para nombres de archivos, pero también puede ser usado sobre cadenas comunes. El usuario promedio puede encontrarse familiarizado con patrones de intérprete de comandos, o, por lo menos en su forma más simple, con los comodines *'?'* y *'*'*, así que el uso de **fnmatch()** en lugar de **ereg()** o **preg_match()** para el procesamiento de expresiones de búsqueda puede ser mucho más conveniente para usuarios no-programadores.

Ejemplo 1. Verificar un nombre de color contra un patrón de comodines de intérprete de comandos.

```php
<?php
if (fnmatch("*gr[ae]y", $color)) {
  echo "alguna forma de gris (gray) ...";
}
?>
```

Aviso
Por ahora, esta función no se encuentra disponible en Windows ni otros sistemas no-compatibles con POSIX.

Vea también **glob()**, **ereg()**, **preg_match()** y la página man Unix sobre *fnmatch(3)* para consultar los nombres de banderas (mientras no estén documentados aquí).

fopen

(PHP 3, PHP 4 , PHP 5)

fopen -- Abre un archivo o URL

Descripción

resource **fopen** (string nombre_archivo, string modo [, bool usar_ruta_inclusion [, resource contexto_z]])

fopen() asocia un recurso con nombre, especificado por *nombre_archivo*, a una secuencia. Si *nombre_archivo* es de la forma "esquema://...", se asume que es una URL y PHP buscará por un gestor de protocolo (también conocido como envoltura) para tal esquema. Si no hay envolturas registradas para ese protocolo, PHP emitirá una noticia para ayudarle a rastrear problemas potenciales en su script, y luego continúa como si *nombre_archivo* indicara un archivo corriente.

Si PHP decide que *nombre_archivo* hace referencia a un archivo local, entonces intentará abrir una secuencia sobre ese archivo. El archivo debe ser asequible para PHP, así que debe asegurarse de que los permisos de acceso del archivo sean los apropiados. Si tiene habilitado safe mode, o open_basedir, pueden aplicarse mayores restricciones.

Si PHP decide que *nombre_archivo* hace referencia a un protocolo registrado, y ese protocolo está registrado como una URL de red, PHP verificará que allow_url_fopen se encuentre habilitado. Si no es así, PHP emitirá una advertencia y la llamada a fopen fallará.

> **Nota:** La lista de protocolos soportados puede encontrarse en Apéndice L. Algunos protocolos (también conocidos como *envolturas*) soportan un *contexto* u opciones `php.ini`. Refiérase a la página específica del protocolo en uso para una lista de opciones que pueden definirse. (Por ejemplo, el valor `php.ini` *user_agent* usado por la envoltura *http*) Para una descripción de los *contextos* y el parámetro *contexto_z*, consulte Referencia CXX, *Funciones de Secuencias*.
>
> **Nota:** Soporte de contexto fue introducido con PHP.5.0.0.

El parámetro *modo* especifica el tipo de acceso que requiere para la secuencia. Puede ser cualquiera de los siguientes valores:

Tabla 1. Una lista de modos posibles para fopen() usando *modo*

modo	Descripción
'r'	Apertura para sólo lectura; ubica el apuntador de archivo al comienzo del mismo.
'r+'	Apertura para lectura y escritura; ubica el apuntador de archivo al comienzo del mismo.
'w'	Apertura para sólo escritura; ubica el apuntador de archivo al comienzo de éste y lo trunca a una longitud de cero. Si el archivo no existe, intenta crearlo.
'w+'	Apertura para lectura y escritura; ubica el apuntador de archivo al comienzo de éste y lo trunca a una longitud cero. Si el archivo no existe, intenta crearlo.
'a'	Apertura para sólo escritura; ubica el apuntador de archivo al final del mismo. Si el archivo no existe, intenta crearlo.
'a+'	Apertura para lectura y escritura; ubica el apuntador de archivo al final del mismo. Si el archivo no existe, intenta crearlo.
'x'	Creación y apertura para sólo escritura; ubica el apuntador de archivo al comienzo de éste. Si el archivo ya existe, la llamada a **fopen()** fallará devolviendo **`FALSE`** y generando un error de nivel **`E_WARNING`**. Si el archivo no existe, intenta crearlo. Esto es ñalente a especificar las banderas *O_EXCL|O_CREAT* en la llamada de sistema *open(2)* interna. Esta opción es soportada en PHP 4.3.2 y versiones posteriores, y sólo funciona con archivos locales.
'x+'	Creación y apertura para lectura y escritura; ubica el apuntador de archivo al comienzo de éste. Si el archivo ya existe, la llamada a **fopen()** fallará devolviendo **`FALSE`** y generando un error de nivel **`E_WARNING`**. Si el archivo no existe, intenta crearlo. Esto es ñalente a especificar las banderas *O_EXCL|O_CREAT* en la llamada de sistema *open(2)* interna. Esta opción es soportada en PHP 4.3.2 y versiones posteriores, y sólo funciona con archivos locales.

Nota: Diferentes familias de sistemas operativos tienen diferentes convenciones sobre el final-de-línea. Cuando escribe a un archivo de texto y desea insertar un salto de línea, necesita usar los caracteres correctos de final-de-línea para su sistema operativo. Los sistemas basados en Unix usan *\n* como el caracter de final de línea, los sistemas basados en Windows usan *\r\n* como los caracteres de final de línea, y los sistemas basados en Macintosh usan *\r* como el caracter de final de línea.

Si usa los caracteres de final de línea ñocados cuando crea sus archivos, puede que encuentre que otras aplicaciones que abren esos archivos lucirán "extraño".

Windows ofrece una bandera de traducción de modo-texto (*'t'*) la cual traducirá transparentemente *\n* a *\r\n* cuando trabaje con el archivo. En contraste, también puede usar *'b'* para forzar el modo binario, el cual no traduce sus datos. Para usar éstas banderas, indique *'b'* o *'t'* como el último caracter del parámetro *modo*.

El modo de traducción predeterminado depende de la SAPI y la versión de PHP que usa, así que es recomendable especificar siempre la bandera apropiada por razones de portabilidad. Debería usar el modo *'t'* si trabaja con archivos de texto-plano y usa *\n* para delimitar los finales de línea en su script, pero espera que sus archivos sean legibles en aplicaciones como el bloc de notas. Debería usar el modo *'b'* en todos los demás casos.

Si no especifica la bandera 'b' cuando trabaja con archivos binarios, puede experimentar problemas con sus datos, incluyendo archivos de imágen corruptos y problemas extraños con los caracteres *\r\n*.

*Por razones de portabilidad, es bastante recomendable que siempre usa la bandera 'b' cuando abre archivos con **fopen()**.*

De nuevo, por razones de portabilidad, es asimismo muy recomendable que re-escriba el código que use o dependa del modo 't' para que use los finales de línea correctos y el modo 'b' en su lugar.

A partir de PHP 4.3.2, el modo predeterminado es definir binario para todas las plataformas que distinguen entre modos binario y de texto. Si tiene problemas con sus scripts después de actualizarse, intente usar la bandera 't' como una solución temporal hasta que haya hecho su script más portable como se menciona más arriba.

El tercer parámetro, opcional, *usar_ruta_inclusion* puede definirse como '1' o **TRUE** si desea buscar por el archivo en include_path, también.

Si la apertura falla, la función devuelve **FALSE** y se genera un error de nivel **E_WARNING**. Es posible usar @ para suprimir tal advertencia.

Ejemplo 1. Ejemplos de fopen()

```
<?php
$gestor = fopen("/home/rasmus/archivo.txt", "r");
$gestor = fopen("/home/rasmus/archivo.gif", "wb");
$gestor = fopen("http://www.example.com/", "r");
$gestor = fopen("ftp://usuario:contrasenya@example.com/un_archivo.txt", "w");
?>
```

Si experimenta problemas con la lectura y escritura sobre archivos, y está usando la versión tipo módulo de servidor de PHP, recuerde asegurarse de que los archivos y directorios que está usando sean asequibles para el proceso del servidor.

En la plataforma Windows, tenga cuidado de escapar cualquier barra invertida usada en la ruta al arcivo, o use barras hace adelante.

```
<?php
$gestor = fopen("c:\\datos\\info.txt", "r");
?>
```

Aviso
Cuando se usa SSL, Microsoft IIS violara el protocolo, cerrando la conexion sin mandar un indicador close_notify. PHP avisara de esto con este mensaje "SSL: Fatal Protocol Error", cuando llegue al final de los datos. Una solucion a este problema es bajar el nivel de aviso de errores del sistema para que no incluya advertencias. PHP 4.3.7 y versiones posteriores detectan servidores IIS con este problema y suprime la advertencia. Si usais la funcion **fsockopen()** para crear un socket ssl://, tendreis que suprimir la advertencia explicitamente.

Nota: Cuando safe-mode (modo-seguro) está activado, PHP comprueba si los directorios que va a utilizar tienen la misma UID que el script que está siendo ejecutado.

Vea también Apéndice L, **fclose()**, **fgets()**, **fread()**, **fwrite()**, **fsockopen()**, **file()**, **file_exists()**, **is_readable()**, **stream_set_timeout()**, y **popen()**.

fpassthru

(PHP 3, PHP 4 , PHP 5)

fpassthru -- Imprime todos los datos restantes en un apuntador de archivo

Descripción

int **fpassthru** (resource gestor)

Lee hasta EOF en el apuntador de archivo dado a partir de la posición actual y escribe los resultados en el búfer de salida.

Si ocurre un error, **fpassthru()** devuelve **`FALSE`**. De lo contrario, **fpassthru()** devuelve el número de caracteres leídos desde *gestor* y pasados a la salida.

El puntero de fichero debe de ser valido y debe de apuntar a un fichero abierto con exito por **fopen()** o **fsockopen()**.

Puede que necesite llamar **rewind()** para restablecer el apuntador de archivo al comienzo de éste si ya ha escrito datos en el archivo. El archivo es cerrado cuando **fpassthru()** ha terminado de leerlo (causando que *gestor* se vuelva inútil).

Si tan sólo quiere volcar el contenido de un archivo al búfer de salida, sin modificarlo o buscar un desplazamiento en particular, puede que quiera usar la función **readfile()**, la cual le ahorra la llamada a **fopen()**.

> **Nota:** Cuando se usa **fpassthru()** sobre un archivo binario en sistemas Windows, debe asegurarse de abrir el archivo en modo binario, agregando el valor *b* al modo usado en la llamada a **fopen()**.
>
> Es recomendable usar la bandera *b* cuando trate con archivos binarios, incluso si su sistema no lo requiere, de modo que sus scripts sean más portables.

Ejemplo 1. Uso de fpassthru() con archivos binarios

```
<?php

// abrir el archivo en modo binario
$nombre = ".\public\dev\img\ok.png";
$aa = fopen($nombre, 'rb');

// enviar las cabeceras correctas
header("Content-Type: image/png");
header("Content-Length: " . filesize($nombre));

// volcar la imagen y detener el script
fpassthru($aa);
exit;

?>
```

Vea también **readfile()**, **fopen()**, **popen()**, y **fsockopen()**

fputcsv

(no version information, might be only in CVS)

fputcsv -- Format line as CSV and write to file pointer

Description

int **fputcsv** (resource handle [, array fields [, string delimiter [, string enclosure]]])

fputcsv() formats a line (passed as a *fields* array) as CSV and write it to the specified file *handle*. Returns the length of the written string, or **FALSE** on failure.

The optional *delimiter* parameter sets the field delimiter (one character only). Defaults as a comma: ,.

The optional *enclosure* parameter sets the field enclosure (one character only) and defaults to a double quotation mark: ".

Ejemplo 1. fputcsv() example

```
<?php

$list = array (
    'aaa,bbb,ccc,dddd',
    '123,456,789',
    '"aaa","bbb"'
);

$fp = fopen('file.csv', 'w');

foreach ($list as $line) {
    fputcsv($fp, split(',', $line));
}

fclose($fp);
?>
```

> **Nota:** Si sufre problemas con *PHP* no reconociendo los finales de línea cuando lee archivos creados en un Macintosh (o leyendo archivos sobre uno), puede probar activando la opción de configuración auto_detect_line_endings.

See also **fgetcsv()**.

fputs

fputs -- Alias de **fwrite()**

Descripción

Esta función es un alias de **fwrite()**.

fread

(PHP 3, PHP 4 , PHP 5)

fread -- Lectura de archivos segura con material binario

Descripción

string **fread** (resource gestor, int longitud)

fread() lee hasta *longitud* bytes desde el apuntador de archivo indicado por *gestor*. La lectura se detiene cuando se han leído *longitud* bytes, se alcanza EOF (el final de archivo), o (en el caso de secuencias de red) cuando un paquete se encuentra disponible, aquello que ocurra primero.

```php
<?php
// obtiene el contenido de un archivo en una cadena
$nombre_archivo = "/usr/local/algo.txt";
$gestor = fopen($nombre_archivo, "r");
$contenido = fread($gestor, filesize($nombre_archivo));
fclose($gestor);
?>
```

Aviso
En sistemas que diferencian entre archivos binarios y de texto (es decir, Windows) el archivo debe ser abierto con el valor 'b' incluido en el parámetro de modo de **fopen()**.

```php
<?php
$nombre_archivo = "c:\\archivos\\una_imagen.gif";
$gestor = fopen($nombre_archivo, "rb");
$contenido = fread($gestor, filesize($nombre_archivo));
fclose($gestor);
?>
```

Aviso
Cuando se lee desde secuencias de red o pipes, como es el caso cuando se leen archivos remotos o desde **popen()** y **fsockopen()**, la lectura se detendrá después de que un paquete esté disponible. Esto quiere decir que debe recolectar los datos en segmentos, como se muestra en el ejemplo a continuación.

```php
<?php
$gestor = fopen("http://www.example.com/", "rb");
$contenido = '';
while (!feof($gestor)) {
  $contenido .= fread($gestor, 8192);
}
fclose($gestor);
?>
```

> **Nota:** Si tan solo desea obtener el contenido de un archivo en una cadena, use **file_get_contents()**, ya que tiene un rendimiento mucho mayor que el código anterior.

Vea también **fwrite()**, **fopen()**, **fsockopen()**, **popen()**, **fgets()**, **fgetss()**, **fscanf()**, **file()**, y **fpassthru()**.

fscanf

(PHP 4 >= 4.0.1, PHP 5)

fscanf -- Procesa la entrada desde un archivo de acuerdo a un formato

Descripción

mixed **fscanf** (resource gestor, string formato [, mixed &...])

La función **fscanf()** es similar a **sscanf()**, pero toma su entrada desde un archivo asociado con *gestor* e interpreta la entrada de acuerdo al *formato* especificado, el cual es descrito en la documentación de **sprintf()**. Si sólo se pasan dos parámetros a esta función, los valores procesados serán devueltos como una matriz. De otro modo, si se pasan parámetros opcionales, la función devolverá el número de valores asignados. Los parámetros opcionales deben ser pasados por referencia.

Cualquier espacio en blanco en la cadena de formato crea una correspondencia con cualquier espacio en blanco en la secuencia de entrada. Esto quiere decir que incluso una tabulación \t en la

cadena de formato puede coincidir con un caracter de espacio sencillo en la secuencia de entrada.

Ejemplo 1. Ejemplo de fscanf()

```
<?php
$gestor = fopen("usuarios.txt","r");
while ($info_usuario = fscanf($gestor, "%s\t%s\t%s\n")) {
    list ($nombre, $profesion, $cod_pais) = $info_usuario;
    //... hacer algo con los valores
}
fclose($gestor);
?>
```

Ejemplo 2. Contenido de usuarios.txt

```
javier  argonauta       pe
hiroshi escultor        jp
robert  desempleado     us
luigi   florista        it
```

> **Nota:** Antes de PHP 4.3.0, el máximo número de caracteres leídos desde el archivo era 512 (o hasta el primer \n, lo que primero ocurriera). A partir de PHP 4.3.0 se leerán y analizarán líneas de longitudes arbitrariamente grandes.

Vea también **fread()**, **fgets()**, **fgetss()**, **sscanf()**, **printf()**, y **sprintf()**.

fseek

(PHP 3, PHP 4 , PHP 5)

fseek -- Realiza una búsqueda sobre un apuntador de archivo

Descripción

int **fseek** (resource gestor, int desplazamiento [, int desde])

Establece el indicador de posición para el archivo referenciado por *gestor*. La nueva posición, medida en bytes desde el comienzo del archivo, so obtiene al sumar *desplazamiento* con la posición especificada por *desde*, cuyos valores se definen como se indica a continuación:

SEEK_SET - Define la posición igual a *desplazamiento* bytes.
SEEK_CUR - Define la posición como la posición actual más *desplazamiento*.
SEEK_END - Define la posición como el final-de-archivo más *desplazamiento*. (Para moverse a una posición anterior al final-de-archivo, es necesario pasar un valor negativo en *desplazamiento*.)
Si no se especifica *desde*, se asume que sea **SEEK_SET**.

De tener éxito, la función devuelve 0; de lo contrario devuelve -1. Note que realizar una reubicación más allá del final de archivo no se considera un error.

Ejemplo 1. Ejemplo de fseek()

```php
<?php

$da = fopen('algun_archivo.txt');

// leer datos
$datos = fgets($da, 4096);

// moverse de vuelta al comienzo del archivo
// igual que rewind($da);
fseek($da, 0);

?>
```

Puede que no sea posible usar la función sobre apuntadores de archivo devueltos por **fopen()** si usan los formatos "http://" o "ftp://". **fseek()** produce también resultados indefinidos para secuencias de adición (abiertas con la bandera "a").

> **Nota:** El argumento *desde* fue agregado después de PHP 4.0.0.

Vea también **ftell()** y **rewind()**.

fstat

(PHP 4 , PHP 5)

fstat -- Obtiene información sobre un archivo usando un apuntador de archivo abierto

Descripción

array **fstat** (resource gestor)

Reúne las estadísticas del archivo abierto por el apuntador a archivo *gestor*. Esta función es similar a la función **stat()**, excepto que opera sobre un apuntador de archivo abierto en lugar de un nombre de archivo.

Devuelve una matriz con las estadísticas del archivo; el formato de la matriz es descrito en la página del manual sobre **stat()**.

Ejemplo 1. Ejemplo de fstat()

```php
<?php

// abrir un archivo
$da = fopen("/etc/passwd", "r");

// reunir estadisticas
$fstat = fstat($da);

// cerrar el archivo
fclose($da);

// imprimir solo la parte asociativa
print_r(array_slice($fstat, 13));

?>
```

esto producirá :

```
Array
(
    [dev] => 771
    [ino] => 488704
    [mode] => 33188
    [nlink] => 1
    [uid] => 0
    [gid] => 0
    [rdev] => 0
    [size] => 1114
    [atime] => 1061067181
    [mtime] => 1056136526
    [ctime] => 1056136526
    [blksize] => 4096
    [blocks] => 8
)
```

Nota: Los resultados de esta función son guardados. Consultar **clearstatcache()** para más detalles.

Nota: Esta funcion no funcionara con ficheros remotos ya que el fichero a examinar tiene que estar disponible desde el sistema de ficheros del servidor.

ftell

(PHP 3, PHP 4 , PHP 5)

ftell -- Indica la posición de lectura/escritura del apuntador de archivo

Descripción

int **ftell** (resource gestor)

Devuelve la posición del apuntador de archivo indicado por *gestor*; es decir, su desplazamiento al interior de la secuencia de archivo.

Si ocurre un error, devuelve **FALSE**.

El apuntador de archivo debe ser válido, y debe apuntar a un archivo abierto satisfactoriamente por **fopen()** o **popen()**. **ftell()** entrega resultados indefinidos para secuencias de adición (abiertas con la bandera "a").

Ejemplo 1. Ejemplo de ftell()

```
<?php

// abre un archivo y lee algunos datos
$da = fopen("/etc/passwd", "r");
$datos = fgets($da, 12);

// en donde estamos ?
echo ftell($da); // 11

fclose($da);

?>
```

Vea también **fopen()**, **popen()**, **fseek()**, and **rewind()**.

ftruncate

(PHP 4 , PHP 5)

ftruncate -- Trunca un archivo a la longitud dada

Descripción

bool **ftruncate** (resource gestor, int tamanyo)

Toma el apuntador de archivo, *gestor*, y trunca el archivo a la longitud *tamanyo*. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

> **Nota:** En versiones anteriores a PHP 4.3.3, **ftruncate()** devuelve un valor **integer** de 1 cuando tiene éxito, en lugar del valor **boolean** **TRUE**.

Vea también **fopen()** y **fseek()**.

fwrite

(PHP 3, PHP 4 , PHP 5)

fwrite -- Escritura sobre archivos, segura con material binario

Descripción

int **fwrite** (resource gestor, string cadena [, int longitud])

fwrite() escribe los contenidos de *cadena* a la secuencia de archivo apuntada por *gestor*. Si el argumento *longitud* es entregado, la escritura se detendrá después de que *longitud* bytes hayan sido escritos, o al alcanzar el final de *cadena*, aquello que ocurra primero.

fwrite() devuelve el número de bytes escritos, o **FALSE** en caso de fallo.

Note que si se utiliza el argumento *longitud*, entonces la opción de configuración magic_quotes_runtime será ignorada y no se eliminarán caracteres de barra desde la *cadena*.

> **Nota:** En los sistemas que diferencian entre archivos binarios y de texto (es decir, Windows) el archivo debe ser abierto incluyendo el valor 'b' en el parámetro de modo de **fopen()**.

Ejemplo 1. Un ejemplo sencillo de fwrite

```
<?php
$nombre_archivo = 'prueba.txt';
$contenido = "Agregar esto al archivo\n";

// Asegurarse primero de que el archivo existe y puede escribirse sobre el.
if (is_writable($nombre_archivo)) {

    // En nuestro ejemplo estamos abriendo $nombre_archivo en modo de adicion.
    // El apuntador de archivo se encuentra al final del archivo, asi que
    // alli es donde ira $contenido cuando llamemos fwrite().
    if (!$gestor = fopen($nombre_archivo, 'a')) {
         echo "No se puede abrir el archivo ($nombre_archivo)";
         exit;
    }

    // Escribir $contenido a nuestro arcivo abierto.
    if (fwrite($gestor, $contenido) === FALSE) {
        echo "No se puede escribir al archivo ($nombre_archivo)";
        exit;
    }

    echo "&Eacute;xito, se escribi&oacute; ($contenido) al archivo ($nombre_archivo)";

    fclose($gestor);

} else {
    echo "No se puede escribir sobre el archivo $nombre_archivo";
}
?>
```

Vea también **fread()**, **fopen()**, **fsockopen()**, **popen()**, y **file_put_contents()**.

glob

(PHP 4 >= 4.3.0, PHP 5)

glob -- Encontrar nombres de ruta coincidentes con un patrón

Descripción

array **glob** (string patrón [, int banderas])

La función **glob()** realiza una búsqueda por todos los nombres de ruta que coincidan con *patrón* de acuerdo a las reglas usadas por la función glob() de la biblioteca de C, las cuales son muy similares a las reglas usadas por intérpretes de comandos comunes. No se realiza expansión de tildes o parámetros.

Devuelve una matriz que contiene los archivos/directorios coincidentes, o **FALSE** si ocurre un error.

Banderas válidas:

- **GLOB_MARK** - Agrega una barra a cada elemento devuelto
- **GLOB_NOSORT** - Devuelve los archivos como aparecen en el directorio (sin ordenar)
- **GLOB_NOCHECK** - Devuelve el patrón de búsqueda si no se han encontrado archivos coincidentes
- **GLOB_NOESCAPE** - Las barras invertidas no indican metacaracteres

- **GLOB_BRACE** - Expande {a,b,c} para que coincida con 'a', 'b', o 'c'
- **GLOB_ONLYDIR** - Devuelve únicamente entradas de directorios que coinciden con el patrón

  Nota: Antes de PHP 4.3.3 **GLOB_ONLYDIR no** estaba disponible en windows y otros sistemas que no usan la biblioteca de C GNU.

Ejemplo 1. Modo conveniente de reemplazar opendir() y amigos con glob().

```
<?php
foreach (glob("*.txt") as $nombre_archivo) {
    echo "$nombre_archivo tam " . filesize($nombre_archivo) . "\n";
}
?>
```

La salida se verá algo como:

```
funclist.txt size 44686
funcsummary.txt size 267625
quickref.txt size 137820
```

Nota: Esta funcion no funcionara con ficheros remotos ya que el fichero a examinar tiene que estar disponible desde el sistema de ficheros del servidor.

Vea también **opendir()**, **readdir()**, **closedir()**, y **fnmatch()**.

is_dir

(PHP 3, PHP 4 , PHP 5)

is_dir -- Indica si el nombre de archivo es un directorio

Descripción

bool **is_dir** (string nombre_archivo)

Devuelve **TRUE** si el archivo con el nombre dado existe y es un directorio. Si *nombre_archivo* es un nombre de archivo relativo, éste será analizado relativo al directorio de trabajo actual.

Nota: Los resultados de esta función son guardados. Consultar **clearstatcache()** para más detalles.

Ejemplo 1. Ejemplo de is_dir()

```
<?
var_dump(is_dir('un_archivo.txt')) . "\n";
var_dump(is_dir('directorio_falso/abc')) . "\n";

var_dump(is_dir('..')); //un directorio arriba
?>
```

El ejemplo anterior produciría la salida:

```
bool(false)
bool(false)
bool(true)
```

Sugerencia: A partir de PHP 5.0.0, esta funcion tambien puede usarse con *algunas* URL como nombre de fichero. Consultar Apéndice L, para obtener una lista con soporte para la funcionalidad **stat()**.

Vea también **chdir()**, dir, **opendir()**, **is_file()** y **is_link()**.

is_executable

(PHP 3, PHP 4 , PHP 5)

is_executable -- Indica si el archivo es ejecutable

Descripción

bool **is_executable** (string nombre_archivo)

Devuelve **TRUE** si el archivo con el nombre dado existe y es ejecutable.

is_executable() apareció en Windows en la versión 5.0.0 de PHP.

Ejemplo 1. Ejemplo de is_executable()

```
<?php

$archivo = '/home/vincent/algun_archivo.sh';

if (is_executable($archivo)) {
    echo $archivo.' es ejecutable';
} else {
    echo $archivo.' no es ejecutable';
}

?>
```

> **Nota:** Los resultados de esta función son guardados. Consultar **clearstatcache()** para más detalles.
>
> **Sugerencia:** A partir de PHP 5.0.0, esta funcion tambien puede usarse con *algunas* URL como nombre de fichero. Consultar Apéndice L, para obtener una lista con soporte para la funcionalidad **stat()**.

Vea también **is_file()** y **is_link()**.

is_file

(PHP 3, PHP 4 , PHP 5)

is_file -- Indica si el archivo es un archivo regular

Descripción

bool **is_file** (string nombre_archivo)

Devuelve **TRUE** si el archivo con el nombre dado existe y es un archivo regular.

> **Nota:** Los resultados de esta función son guardados. Consultar **clearstatcache()** para más detalles.
>
> **Sugerencia:** A partir de PHP 5.0.0, esta funcion tambien puede usarse con *algunas*

URL como nombre de fichero. Consultar Apéndice L, para obtener una lista con soporte para la funcionalidad **stat()**.

Vea también **is_dir()** y **is_link()**.

is_link

(PHP 3, PHP 4 , PHP 5)

is_link -- Indica si el archivo es un enlace simbólico

Descripción

bool **is_link** (string nombre_archivo)

Devuelve **TRUE** si el archivo con el nombre dado existe y es un enlace simbólico.

Nota: Los resultados de esta función son guardados. Consultar **clearstatcache()** para más detalles.

Sugerencia: A partir de PHP 5.0.0, esta funcion tambien puede usarse con *algunas* URL como nombre de fichero. Consultar Apéndice L, para obtener una lista con soporte para la funcionalidad **stat()**.

Vea también **is_dir()**, **is_file()**, y **readlink()**.

is_readable

(PHP 3, PHP 4 , PHP 5)

is_readable -- Indica si es posible leer el archivo

Descripción

bool **is_readable** (string nombre_archivo)

Devuelve **TRUE** si el archivo existe y es posible leerlo.

Tenga en cuenta que PHP puede estar accediendo al archivo bajo el id de usuario bajo el que corre el servidor web (usualmente 'nobody'). Las limitaciones dol modo seguro no son tomadas en cuenta.

Nota: Los resultados de esta función son guardados. Consultar **clearstatcache()** para más detalles.

Sugerencia: A partir de PHP 5.0.0, esta funcion tambien puede usarse con *algunas* URL como nombre de fichero. Consultar Apéndice L, para obtener una lista con soporte para la funcionalidad **stat()**.

Vea también **is_writable()**, **file_exists()**, y **fgets()**.

is_uploaded_file

(PHP 3>= 3.0.17, PHP 4 >= 4.0.3, PHP 5)

is_uploaded_file -- Indica si un archivo fue cargado a través de HTTP POST

Descripción

bool **is_uploaded_file** (string nombre_archivo)

Devuelve **TRUE** si el archivo dado por *nombre_archivo* fue cargado a través de HTTP POST. Esto es útil para ayudar a verificar que un usuario malicioso no ha intentado engañar al script haciéndole trabajar sobre archivos con los que no debería trabajar--por ejemplo, `/etc/passwd`.

Este tipo de chequeo es especialmente importante si existe alguna posibilidad de que cualquier cosa realizada con archivos cargados pueda revelar sus contenidos al usuario, o incluso a otros usuarios en el mismo sistema.

is_uploaded_file() se encuentra disponible únicamente en versiones de PHP 3 superiores a PHP 3.0.16, y en versiones de PHP 4 superiores a 4.0.2. Si está atrapado usando una versión anterior, puede usar la siguiente función para protegerse:

> **Nota:** El siguiente ejemplo *no* trabajará con versiones de PHP 4 superiores a 4.0.2. Depende en la funcionalidad interna de PHP que fue modificada luego de esa versión.

Ejemplo 1. Ejemplo de is_uploaded_file()

```
<?php
/* Prueba de usuario para verificar un archivo cargado. */
function is_uploaded_file($nombre_archivo)
{
    if (!$archivo_tmp = get_cfg_var('upload_tmp_dir')) {
        $archivo_tmp = dirname(tempnam('', ''));
    }
    $archivo_tmp .= '/' . basename($nombre_archivo);
    /* El usuario puede tener una barra final en php.ini... */
    return (ereg_replace('/+', '/', $archivo_tmp) == $nombre_archivo);
}

/* Asi es como se usa, ya que tampoco se cuenta con
 * move_uploaded_file() en estas versiones antiguas: */
if (is_uploaded_file($HTTP_POST_FILES['archivo_de_usuario'])) {
    copy($HTTP_POST_FILES['archivo_de_usuario'], "/lugar/a/colocar/el/archivo/cargado");
} else {
    echo "Posible ataque de archivo entrante: nombre de archivo '$HTTP_POST_FILES[archi
}
?>
```

Vea también **move_uploaded_file()**, y la sección Gestión de carga de archivos para un ejempo de uso sencillo.

is_writable

(PHP 4 , PHP 5)

is_writable -- Indica si el nombre de archivo es escribible

Descripción

bool **is_writable** (string nombre_archivo)

Devuelve **TRUE** si *nombre_archivo* existe y es escribible. El argumento de nombre de archivo puede ser un nombre de directorio, permitiéndole revisar si un directorio es escribible.

Tenga en mente que PHP puede estar accediendo al archivo bajo el id de usuario con el que corre el servidor web (usualmente 'nobody'). No se tienen en cuenta limitaciones del modo seguro.

> **Nota:** Los resultados de esta función son guardados. Consultar **clearstatcache()** para más detalles.
>
> **Sugerencia:** A partir de PHP 5.0.0, esta funcion tambien puede usarse con *algunas* URL como nombre de fichero. Consultar Apéndice L, para obtener una lista con soporte para la funcionalidad **stat()**.

Vea también **is_readable()**, **file_exists()**, y **fwrite()**.

is_writeable

is_writeable -- Alias de **is_writable()**

Descripción

Esta función es un alias de **is_writable()**.

link

(PHP 3, PHP 4 , PHP 5)

link -- Crea un enlace fuerte

Descripción

int **link** (string target, string link)

link() crea un enlace fuerte.

Ver también **symlink()** para crear enlaces débiles, y **readlink()** junto con **linkinfo()**.

linkinfo

(PHP 3, PHP 4 , PHP 5)

linkinfo -- Consigue información sobre un enlace

Descripción

int **linkinfo** (string path)

linkinfo() da el campo st_dev de la extructura stat de UNIX C devuelto por la llamada al sistema lstat. Esta función se usa para verificar si un enlace (apuntado por *path*) existe realmente (usando el mismo método que la macro S_ISLNK definida en stat.h). Devuelve 0 o **`FALSE`** en caso de error.

Ver también **symlink()**, **link()**, y **readlink()**.

lstat

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

lstat -- Entrega información sobre un archivo o enlace simbólico

Descripción

array **lstat** (string nombre_archivo)

Reúne las estadísticas del archivo o enlace simbólico con el nombre *nombre_archivo*. Esta función es idéntica a la función **stat()**, con la excepción de que si el parámetro *nombre_archivo* es un enlace simbólico, se devuelve el status del enlace simbólico, no el del archivo apuntado por el enlace simbólico.

Consulte la página del manual sobre **stat()** para información sobre la estructura de la matriz que **lstat()** devuelve.

> **Nota:** Los resultados de esta función son guardados. Consultar **clearstatcache()** para más detalles.
>
> **Sugerencia:** A partir de PHP 5.0.0, esta funcion tambien puede usarse con *algunas* URL como nombre de fichero. Consultar Apéndice L, para obtener una lista con soporte para la funcionalidad **stat()**.

Vea también **stat()**.

mkdir

(PHP 3, PHP 4 , PHP 5)

mkdir -- Crea un directorio

Descripción

int **mkdir** (string pathname, int mode)

Trata de crear el directorio especificado por pathname.

Ten en cuenta que debes especifiar el modo como un número octal, lo que significa que debes

anteponerle un 0 al número.

```
mkdir ("/path/to/my/dir", 0700);
```

Devuelve **TRUE** en caso de éxito y **FALSE** en caso de fallo.

Ver también **rmdir()**.

move_uploaded_file

(PHP 4 >= 4.0.3, PHP 5)

move_uploaded_file -- Mueve un archivo cargado a una nueva ubicación

Descripción

bool **move_uploaded_file** (string nombre_archivo, string destino)

Esta función realiza un chequeo para asegurar que el archivo indicado por *nombre_archivo* sea un archivo cargado válido (lo que quiere decir que fue cargado a través del mecanismo de carga HTTP POST de PHP). Si el archivo es válido, será movido al nombre de archivo dado por *destino*.

Si *nombre_archivo* no es un archivo cargado válido, entonces no se tomará ninguna acción, y **move_uploaded_file()** devolverá **FALSE**.

Si *nombre_archivo* es un archivo cargado válido, pero no puede ser movido por alguna razón, no se tomará ninguna acción, y **move_uploaded_file()** devolverá **FALSE**. Adicionalmente, se emitirá una advertencia.

Este tipo de chequeo es especialmente importante si hay algún chance de que cualquier cosa hecha con archivos cargados pueda revelar sus contenidos al usuario, o incluso a otros usuarios en el mismo sistema.

> **Nota:** Cuando safe-mode (modo-seguro) está activado, PHP comprueba si los archivos o directorios que va a utilizar tienen la misma UID que el script que está siendo ejecutado.
>
> **Nota:** La función **move_uploaded_file()** no se ve afectada por las restricciones safe mode normales. Esto no es inseguro ya que **move_uploaded_file()** solo opera con archivos cargados a través de PHP.

Aviso
Si el archivo de destino ya existe, será sobrescrito.

Vea también **is_uploaded_file()**, y la sección Gestión de carga de archivos para un ejemplo de uso simple.

parse_ini_file

(PHP 4 , PHP 5)

parse_ini_file -- Procesar un archivo de configuración

Descripción

array **parse_ini_file** (string nombre_archivo [, bool procesar_secciones])

parse_ini_file() lee el contenido del archivo ini especificado en *nombre_archivo*, y devuelve los parámetros que incluye en una matriz asociativa. Al definir el último parámetro *procesar_secciones* como **TRUE**, recibe una matriz multidimensional, con los nombres de secciones y parámetros incluidos. El valor predeterminado para *procesar_secciones* es **FALSE**

> **Nota:** Esta función no tiene relación alguna con el archivo `php.ini`. Éste ya ha sido procesado al momento de ejecutar su script. Esta función puede ser usada para leer los archivos de configuración de su propia aplicación.
>
> **Nota:** Si un valor en el archivo ini contiene caracteres no-alfanuméricos, éste necesita ser rodeado por comillas dobles (").
>
> **Nota:** A partir de PHP 4.2.1, esta función se ve afectada también por safe mode y open_basedir.
>
> **Nota:** Existen palabras reservadas que no deben ser usadas como claves en archivos ini. Entre estas se encuentran: null, yes, no, true, y false.

La estructura del archivo ini es similar al de `php.ini`.

También pueden procesarse Constantes en el archivo ini, de tal modo que si define una constante como un valor ini antes de ejecutar **parse_ini_file()**, ésta será integrada en los resultados. Solo son evaluados los valores ini. Por ejemplo:

Ejemplo 1. Contenidos de `ejemplo.ini`

```
; Este es un archivo de configuracion de ejemplo
; Los comentarios comienzan con ';', como en php.ini

[primera_seccion]
uno = 1
cinco = 5
animal = PAJARO

[segunda_seccion]
ruta = /usr/local/bin
URL = "http://www.example.com/~nombreusuario"
```

Ejemplo 2. Ejemplo de parse_ini_file()

```
<?php

define('PAJARO', 'Ave Dodo');

// Procesar sin secciones
$matriz_ini = parse_ini_file("ejemplo.ini");
print_r($matriz_ini);

// Procesar con secciones
$matriz_ini = parse_ini_file("ejemplo.ini", true);
print_r($matriz_ini);

?>
```

Produciría:

```
Array
(
    [uno] => 1
    [cinco] => 5
    [animal] => Ave Dodo
    [ruta] => /usr/local/bin
    [URL] => http://www.example.com/~nombreusuario
)
Array
(
    [primera_seccion] => Array
        (
            [uno] => 1
            [cinco] => 5
            [animal] = Ave Dodo
        )

    [segunda_seccion] => Array
        (
            [ruta] => /usr/local/bin
            [URL] => http://www.example.com/~nombreusuario
        )

)
```

Las claves y los nombres de sección que consisten de números son evaluadas como enteros de PHP, de modo que los números que comienzan con 0 son evaluados como valores octales y los números que comienzan con 0x son evaluados como hexadecimales.

pathinfo

(PHP 4 >= 4.0.3, PHP 5)

pathinfo -- Devuelve información sobre la ruta de un archivo

Descripción

array **pathinfo** (string ruta [, int opciones])

pathinfo() devuelve una matriz asociativa que contiene información sobre *ruta*. Los siguientes elementos de la matriz son devueltos: *dirname*, *basename* y *extension*.

Puede especificar cuáles elementos son devueltos con el parámetro opcional *opciones*. Éste se compone de **PATHINFO_DIRNAME**, **PATHINFO_BASENAME** y **PATHINFO_EXTENSION**. Su comportamiento predeterminado es devolver todos los elementos.

Ejemplo 1. Ejemplo de pathinfo()

```
<?php
$partes_ruta = pathinfo('/www/htdocs/index.html');

echo $partes_ruta['dirname'] . "\n";
echo $partes_ruta['basename'] . "\n";
echo $partes_ruta['extension'] . "\n";
?>
```

Produciría:

```
/www/htdocs
index.html
html
```

> **Nota:** Para información sobre la recuperación de información sobre la ruta actual, lea la sección relevante en variables reservadas predefinidas.

Vea también **dirname()**, **basename()**, **parse_url()** y **realpath()**.

pclose

(PHP 3, PHP 4 , PHP 5)

pclose -- Cierra un apuntador de archivo de proceso

Descripción

int **pclose** (resource gestor)

Cierra un apuntador de archivo a un pipe abierto por **popen()**.

El apuntador de archivo debe ser válido, y debe haber sido devuelto por un llamado satisfactorio a **popen()**.

Devuelve el status de terminación del proceso que fue ejecutado.

Vea también **popen()**.

popen

(PHP 3, PHP 4 , PHP 5)

popen -- Abre un apuntador de archivo de proceso

Descripción

resource **popen** (string comando, string modo)

Abre un pipe con un proceso ejecutado al bifurcar el comando dado en el primer parámetro.

Devuelve un apuntador de archivo idéntico al devuelto por **fopen()**, con la excepción de que es unidireccional (puede ser usado sólo para lectura o escritura) y debe ser cerrado con **pclose()**. Este apuntador puede ser usado con **fgets()**, **fgetss()**, y **fwrite()**.

Si ocurre un error, se devuelve **`FALSE`**.

> **Nota:** Si está buscando soporte bidireccional (en dos vías), use **proc_open()**.

Ejemplo 1. Ejemplo de popen()

```
<?php
$gestor = popen("/bin/ls", "r");
?>
```

Si el comando a ser ejecutado no es encontrado, se devuelve un recurso válido. Esto puede parecer extraño, pero tiene sentido; le permite acceder a cualquier mensaje de error devuelto por el intérprete de comandos:

```
<?php
error_reporting(E_ALL);

/* Agregar redireccion de modo que tengamos stderr. */
$gestor = popen('/ruta/hacia/basura 2>&1', 'r');
echo "'$gestor'; " . gettype($gestor) . "\n";
$read = fread($gestor, 2096);
echo $read;
pclose($gestor);
?>
```

> **Nota:** Cuando safe mode esta activado, solamente se pueden ejecutar los programas que se encuentren en safe_mode_exec_dir. Por razones practicas, no se permite el uso de .. en el PATH del programa.

Aviso
Con safe mode activado, todas las palabras que siguan al comando inicial son tratadas como un solo argumento. Asi, *echo y | echo x* se interpreta como *echo "y | echo x"*.

Vea también **pclose()**, **fopen()**, y **proc_open()**.

readfile

(PHP 3, PHP 4 , PHP 5)

readfile -- Imprime un archivo

Descripción

int **readfile** (string nombre_archivo [, bool usar_ruta_inclusion [, resource contexto]])

Lee un archivo y lo escribe al búfer de salida.

Devuelve el número de bytes leídos desde el archivo. Si ocurre un error, **FALSE** es devuelto y a menos que la función sea llamada como @**readfile()**, se genera un mensaje de error.

> **Sugerencia:** Puede usar una URL como nombre de archivo con esta función si los fopen wrappers han sido activados. Consulte **fopen()** para más detalles sobre cómo especificar el nombre de fichero y Apéndice L una lista de protocolos URL soportados

Puede usar el segundo parámetro opcional y definirlo como **TRUE**, si desea buscar por el archivo en include_path, también.

Vea también **fpassthru()**, **file()**, **fopen()**, **include()**, **require()**, **virtual()**, **file_get_contents()**, y Apéndice L.

readlink

(PHP 3, PHP 4 , PHP 5)

readlink -- Devuelve el objetivo de un enlace simbólico

Descripción

string **readlink** (string path)

readlink() hace lo mismo que la función C readlink C y devuelve el contenido del path del enlace simbólico o 0 en caso de error.

Ver también **symlink()**, **readlink()** y **linkinfo()**.

realpath

(PHP 4 , PHP 5)

realpath -- Devuelve el nombre de ruta absoluto simplificado

Descripción

string **realpath** (string ruta)

realpath() expande todos los enlaces simbólicos y revuelve todas las referencias a '/./', '/../' y caracteres '/' extra en la *ruta* de entrada, y devuelve el nombre de ruta absoluto simplificado. La ruta resultante no tendrá enlaces simbólicos, ni componentes '/./' o '/../'.

realpath() devuelve **FALSE** en caso de fallo, p.ej. si el archivo no existe.

Ejemplo 1. Ejemplo de realpath()

```
<?php
$ruta_real = realpath("../../index.php");
?>
```

Vea también **basename()**, **dirname()**, y **pathinfo()**.

rename

(PHP 3, PHP 4 , PHP 5)

rename -- Renombra un archivo o directorio

Descripción

bool **rename** (string nombre_antiguo, string nombre_nuevo [, resource contexto])

Intenta renombrar *nombre_antiguo* a *nombre_nuevo*.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplo 1. Ejemplo con rename()

```
<?php
rename("/tmp/archivo_temp.txt", "/home/usuario/login/docs/mi_archivo.txt");
?>
```

Nota: Antes de PHP 4.3.3, **rename()** no podía renombrar archivos a través de

particiones diferentes bajo sistemas basados en Unix.

Nota: A partir de PHP 5.0.0, **rename()** puede usarse también con *algunas* envolturas URL. Consulte Apéndice L para un listado de las envolturas que soporta **rename()**.

Nota: La envoltura usada en *nombre_antiguo DEBE* coincidir con la envoltura usada en *nombre_nuevo*.

Nota: El parámetro *contexto* fue agregado en PHP 5.0.0.

Vea también **copy()**, **unlink()**, y **move_uploaded_file()**.

rewind

(PHP 3, PHP 4 , PHP 5)

rewind -- Retroceder la posición de un apuntador de archivo

Descripción

bool **rewind** (resource gestor)

Define el indicador de posición de archivo para *gestor* como el comienzo de la secuencia de archivo.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

El apuntador de archivo debe ser válido, y debe apuntar a un archivo abierto satisfactoriamente mediante **fopen()**.

Nota: Si ha abierto el archivo en modo de adición ("a"), cualquier información que escriba al archivo será siempre agregada al final de éste, independientemente de la posición del archivo.

Vea también **fseek()** y **ftell()**.

rmdir

(PHP 3, PHP 4 , PHP 5)

rmdir -- Elimina un directorio

Descripción

int **rmdir** (string dirname)

Trata de eliminar el directorio indicado por pathname. El directorio debe estar vacio, y los permisos relevantes deben permitir esto.

Si ocurre un error, devuelve 0.

Ver también **mkdir()**.

set_file_buffer

set_file_buffer -- Alias de **stream_set_write_buffer()**

Descripción

Esta función es un alias de **stream_set_write_buffer()**.

stat

(PHP 3, PHP 4 , PHP 5)

stat -- Entrega información sobre un archivo

Descripción

array **stat** (string nombre_archivo)

Reúne las estadísticas del archivo con el nombre *nombre_archivo*. Si *nombre_archivo* es un enlace simbólico, las estadísticas son del archivo mismo, no del enlace. **lstat()** es una función idéntica a **stat()**, con la excepción de que utiliza la información de status de los enlaces simbólicos.

En caso de fallos, **stat()** devuelve **`FALSE`**. También arrojará una advertencia.

Devuelve una matriz con las estadísticas del archivo, con los siguientes elementos. Esta matriz comienza en cero. Además de devolver estos atributos en una matriz numérica, éstos pueden ser accesados con índices asociativos, como se nota al lado de cada parámetro; esta alternativa está disponible a partir de PHP 4.0.6:

Tabla 1. Formato de resultados de stat() y fstat()

Numérico	Asociativo (desde PHP 4.0.6)	Descripción
0	dev	número de dispositivo
1	ino	número de inode
2	mode	modo de protección inode
3	nlink	número de enlaces
4	uid	id de usuario del dueño
5	gid	id de grupo del dueño
6	rdev	tipo de dispositivo, si hay dispositivo inode *
7	size	tamaño en bytes
8	atime	hora del último acceso (marca de tiempo Unix)
9	mtime	hora de la última modificación (marca de tiempo Unix)

Numérico	Asociativo (desde PHP 4.0.6)	Descripción
10	ctime	hora del último cambio del inode (marca de tiempo Unix)
11	blksize	tamaño de bloque de E/S del sistema de archivos *
12	blocks	número de bloques reservados

* - válido únicamente en sistemas que soportan el tipo st_blksize--los demás sistemas (como Windows) devuelven -1.

> **Nota:** Los resultados de esta función son guardados. Consultar **clearstatcache()** para más detalles.
>
> **Sugerencia:** A partir de PHP 5.0.0, esta funcion tambien puede usarse con *algunas* URL como nombre de fichero. Consultar Apéndice L, para obtener una lista con soporte para la funcionalidad **stat()**.

Vea también **lstat()**, **fstat()**, **filemtime()**, y **filegroup()**.

symlink

(PHP 3, PHP 4 , PHP 5)

symlink -- Crea un enlace simbólico

Descripción

int **symlink** (string target, string link)

symlink() crea un enlace simbólico del objetivo *target* con el nombre especificado por *link*.

Ver también **link()** para crear enlaces fuertes, y **readlink()** junto con **linkinfo()**.

tempnam

(PHP 3, PHP 4 , PHP 5)

tempnam -- Crear un archivo con un nombre único

Descripción

string **tempnam** (string dir, string prefijo)

Crea un archivo con un nombre único en el directorio especificado. Si el directorio no existe, **tempnam()** puede generar un archivo en el directorio temporal del sistema, y devolver su nombre.

En versiones anteriores a PHP 4.0.6, el comportamiento de la función **tempnam()** dependía del sistema. En Windows, la variable de entorno TMP sobrescribía el parámetro *dir*, en Linux la variable TMPDIR tiene precedencia, mientras que SVR4 siempre usa su parámetro *dir* si el directorio al que apunta existe. Consulte la documentación de su sistema sobre la función tempnam

(3) si necesita mayor claridad al respecto.

> **Nota:** Si PHP no puede crear un archivo en el parámetro *dir* especificado, pasa a usar el valor predeterminado del sistema.

Devuelve el nuevo nombre de archivo temporal, o **FALSE** en caso de fallo.

Ejemplo 1. Ejemplo de tempnam()

```
<?php
$nombre_temp = tempnam("/tmp", "FOO");

$gestor = fopen($nombre_temp, "w");
fwrite($gestor, "escribiendo al archivo temporal");
fclose($gestor);

// haga algo aqui

unlink($nombre_temp);
?>
```

> **Nota:** El comportamiento de esta función cambió en 4.0.3. El archivo temporal es creado también para evitar una condición de carrera en donde el archivo puede aparecer en el sistema de archivos entre el periodo en que se genera la cadena y antes de que el script se dedique a crear el archivo. Note que necesita eliminar el archivo en caso de que no lo necesite más, ya que ésto no se hace automáticamente

Vea también **tmpfile()** y **unlink()**.

tmpfile

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

tmpfile -- Crea un archivo temporal

Descripción

resource **tmpfile** (void)

Crea un archivo temporal con un nombre único en modo de escritura, devolviendo un gestor de archivo similar al que devuelve **fopen()**. El archivo es eliminado automáticamente cuando sea cerrado (usando **fclose()**), o cuando el script finalice.

Para más detalles, consulte la documentación de su sistema sobre la función *tmpfile(3)*, así como el archivo de cabecera `stdio.h`.

Ejemplo 1. Ejemplo de tmpfile()

```
<?php
$temp = tmpfile();
fwrite($temp, "escribiendo sobre el archivo temporal");
fclose($temp); // esto elimina el archivo
?>
```

Vea también **tempnam()**.

touch

(PHP 3, PHP 4 , PHP 5)

touch -- Establece la hora de acceso y modificación de un archivo

Descripción

bool **touch** (string nombre_archivo [, int hora [, int hora_acceso]])

Intenta establecer la hora de acceso y modificación del archivo con el nombre dado al valor indicado por el parámetro hora. Si la opción *hora* no se especifica, se usa la hora actual. Esto es ñalente a lo que hace utime (algunas veces conocido como utimes). Si la tercera opción *hora_acceso* se especifica, la hora de acceso del nombre de archivo dado se establece al valor de *hora_acceso*. Note que la hora de acceso siempre se modifica, inpedendientemente del número de parámetros.

Si el archivo no existe, éste es creado. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplo 1. Ejemplo de touch()

```
<?php
if (touch($NombreArchivo)) {
    echo "La hora de modificaci&oacute;n de $NombreArchivo ha sido modificada
          a la fecha y hora de hoy";
} else {
    echo "Lamentablemente no fue posible cambiar la hora de
          modificaci&oacute;n de $NombreArchivo";
}
?>
```

umask

(PHP 3, PHP 4 , PHP 5)

umask -- Cambia la umask actual

Descripción

int **umask** (int mask)

umask() fija las umask PHP con la mascara & 0777 y y devuelve la antigua umask. Cuando PHP se está usando como un módulo del servidor, la umask se restaura cuando cada petición es finalizada.

umask() sin argumentos sólamente devuelve la umask actual.

unlink

(PHP 3, PHP 4 , PHP 5)

unlink -- Elimina un archivo

Descripción

bool **unlink** (string nombre_archivo [, resource contexto])

Elimina *nombre_archivo*. Es similar a la función unlink() de C en Unix. Devuelve **TRUE** si todo se

llevó a cabo correctamente, **FALSE** en caso de fallo.

> **Nota:** A partir de PHP 5.0.0, la función **unlink()** puede ser usada también con *algunas* envolturas de URL. Refiérase a Apéndice L para consultar un listado de las envolturas con soporte para **unlink()**.

> **Nota:** El parámetro *contexto* fue agregado a partir de PHP 5.0.0.

Vea también **rmdir()** para la eliminación de directorios.

XXXVIII. FriBiDi Functions

Introducción

FriBiDi is a free implementation of the Unicode Bidirectional Algorithm.

Requirimientos

You must download and install the FriBiDi package.

Instalación

Esta extension PECL no esta ligada a PHP. Mas informacion sobre nuevos lanzamientos, descargas ficheros de fuentes, informacion sobre los responsables asi como un 'CHANGELOG', se puede encontrar aqui: http://pecl.php.net/package/fribidi.

In order to use these functions you must compile PHP with Fribidi support by using the *--with-fribidi[=DIR]* configure option.

Windows users will enable `php_fribidi.dll` inside of `php.ini` in order to use these functions. Podeis descargar esta DLL de las extensiones PECL desde la pagina PHP Downloads o desde http://snaps.php.net/.

Configuración en tiempo de ejecución

Tipos de recursos

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

FRIBIDI_CHARSET_UTF8 (integer)

FRIBIDI_CHARSET_8859_6 (integer)

FRIBIDI_CHARSET_8859_8 (integer)

FRIBIDI_CHARSET_CP1255 (integer)

FRIBIDI_CHARSET_CP1256 (integer)

FRIBIDI_CHARSET_ISIRI_3342 (integer)

Tabla de contenidos

fribidi_log2vis

(PHP 4 >= 4.0.4)

fribidi_log2vis -- Convert a logical string to a visual one

Description

string **fribidi_log2vis** (string str, string direction, int charset)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

XXXIX. Funciones FTP

Introducción

Las funciones en esta extensión implementan acceso de cliente a servidores de archivos que entiendan el Protocolo de Transferencia de Archivos (FTP, por sus siglas en Inglés), tal y como se define en http://www.faqs.org/rfcs/rfc959. Esta extensión tiene como propósito el acceso detallado a un servidor FTP, brindando un amplio rango de control al script que se encuentre ejecutando. Si sólo desea leer desde un archivo o escribir sobre un archivo en un servidor FTP, considere el uso de la envoltura *ftp://* con las funciones del sistema de archivos, medio que provee una interfaz más simple e intuitiva.

Requirimientos

No se necesitan bibliotecas externas para construir esta extensión

---

Instalación

Para usar las funciones FTP con su configuración PHP, debe agregar la opción *--enable-ftp* cuando instale PHP 4, o *--with-ftp* cuando use PHP 3.

La versión para Windows de *PHP* tiene soporte nativo para esta extensión. No se necesita cargar ninguna extensión adicional para usar estas funciones.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

Esta extensión usa un tipo de recurso, que es el identificador de enlace de la conexión FTP devuelto por **ftp_connect()**.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

`FTP_ASCII` (**integer**)

`FTP_TEXT` (**integer**)

`FTP_BINARY` (**integer**)

`FTP_IMAGE` (**integer**)

`FTP_TIMEOUT_SEC` (**integer**)

Vea **ftp_set_option()** para más información.

Las siguientes constantes fueron introducidas en PHP 4.3.0.

`FTP_AUTOSEEK` (**integer**)

Vea **ftp_set_option()** para más información.

FTP_AUTORESUME (integer)

Determinar automáticamente la posición de continuación y la posición de comienzo para peticiones GET y PUT (funciona únicamente si FTP_AUTOSEEK está habilitado)

FTP_FAILED (integer)

La transferencia asincrónica ha fallado

FTP_FINISHED (integer)

La transferencia asincrónica ha terminado

FTP_MOREDATA (integer)

La transferencia asincrónica está aun activa

Ejemplos

Ejemplo 1. Ejemplo de FTP

```
<?php
// establecer una conexion basica
$id_con = ftp_connect($servidor_ftp);

// inicio de sesion con nombre de usuario y contrasenya
$resultado_login = ftp_login($id_con, $nombre_usuario_ftp, $contrasenya_ftp);

// chequear la conexion
if ((!$id_con) || (!$resultado_login)) {
        echo "&iexcl;La conexi&oacute;n FTP ha fallado!";
        echo "Se ha intentado la conexion con $servidor_ftp para el " .
             "usuario $nombre_usuario_ftp";
        exit;
    } else {
        echo "Conectado con $servidor_ftp, para el usuario $nombre_usuario_ftp";
    }

// cargar el archivo
$carga = ftp_put($id_con, $archivo_destino, $archivo_fuente, FTP_BINARY);

// chequear el status de la carga
if (!$carga) {
        echo "&iexcl;La carga FTP ha fallado!";
    } else {
        echo "Se ha cargado $archivo_fuente a $servidor_ftp como $archivo_destino";
    }

// cierra la secuencia FTP
ftp_close($id_con);
?>
```

Tabla de contenidos

ftp_exec -- Solicita la ejecución de un programa en el servidor FTP
ftp_fget -- Descarga un archivo desde el servidor FTP y lo guarda en un archivo abierto
ftp_fput -- Carga un archivo abierto al servidor FTP
ftp_get_option -- Recupera varios comportamientos de tiempo de ejecución de la secuencia FTP actual
ftp_get -- Descarga un archivo desde el servidor FTP
ftp_login -- Comienza la sesion en una conexión FTP
ftp_mdtm -- Devuelve la última hora de modificación del archivo dado
ftp_mkdir -- Crea un directorio
ftp_nb_continue -- Continúa recuperando/enviando un archivo (modo no-bloqueo)
ftp_nb_fget -- Recupera un archivo desde el servidor FTP y lo escribe sobre un archivo abierto (modo no-bloqueo)
ftp_nb_fput -- Almacena un archivo desde un archivo abierto en el servidor FTP (modo no-bloqueo)
ftp_nb_get -- Recupera un archivo desde el servidor FTP y lo escribe sobre un archivo local (modo no-bloqueo)
ftp_nb_put -- Almacena un archivo en el servidor FTP (modo no-bloqueo)
ftp_nlist -- Devuelve una lista de archivos en el directorio dado
ftp_pasv -- Activa o desactiva el modo pasivo.
ftp_put -- Carga un archivo al servidor FTP
ftp_pwd -- Devuelve el nombre del directorio actual
ftp_quit -- Cierra una conexión FTP
ftp_raw -- Envía un comando arbitrario a un servidor FTP
ftp_rawlist -- Devuelve una lista detallada de archivos en el directorio dado
ftp_rename -- Renombra un archivo en el servidor FTP
ftp_rmdir -- Borra un directorio
ftp_set_option -- Establecer varias opciones FTP de tiempo de ejecución
ftp_site -- Envía un comando SITE al servidor
ftp_size -- Devuelve el tamaño del archivo dado
ftp_ssl_connect -- Abre una conexión segura SSL-FTP
ftp_systype -- Devuelve el identificador de tipo de sistema del servidor FTP remoto.

ftp_alloc

(PHP 5)

ftp_alloc -- Reserva espacio para que un archivo sea cargado

Descripción

bool **ftp_alloc** (resource secuencia_ftp, int tam_archivo [, string &resultado])

Envía un comando ALLO al servidor FTP remoto para ubicar *tam_archivo* bytes de espacio. Devuelve **`TRUE`** en caso de tener éxito, o **`FALSE`** en caso de fallo.

> **Nota:** Muchos servidores FTP no ofrecen soporte para este comando. Estos servidores pueden devolver un código de fallo (**`FALSE`**) que indica que el comando no es soportado, o un código de éxito (**`TRUE`**) para indicar que la pre-ubicación no es necesaria y el cliente debe continuar como si la operación hubiera sido exitosa. Debido a esto, puede ser mejor reservar esta función para servidores que requieren la preubicación explícitamente.

Una representación textual de la respuesta de los servidores será devuelta por referencia en *resultado* si se entrega la variable.

Ejemplo 1. Ejemplo de ftp_alloc()

```
<?php

$archivo = "/home/usuario/miarchivo";

/* conexion con el servidor */
$id_con = ftp_connect('ftp.example.com');
$resultado_login = ftp_login($id_con, 'anonymous', 'usuario@example.com');

if (ftp_alloc($id_con, filesize($archivo), $resultado)) {
  echo "El espacio fue reservado satisfactoriamente en el servidor. Enviando $archivo.\
  ftp_put($id_con, '/incomming/miarchivo', $archivo, FTP_BINARY);
} else {
  echo "No fue posible reservar espacio en el servidor. El servidor dijo: $resultado\n"
}

ftp_close($id_con);

?>
```

Vea también: **ftp_put()**, y **ftp_fput()**.

ftp_cdup

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

ftp_cdup -- Cambia al directorio padre

Descripción

int **ftp_cdup** (int ftp_stream)

Si tiene éxito, devuelve **TRUE**. En caso de error, devuelve **FALSE**.

Cambia al directorio padre.

ftp_chdir

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

ftp_chdir -- Cambia de directorio en un servidor FTP

Descripción

int **ftp_chdir** (int ftp_stream, string directory)

Si tiene éxito, devuelve **TRUE**. En caso de error, devuelve **FALSE**.

Cambia al directorio especificado por el parámetro *directory*.

ftp_chmod

(PHP 5)

ftp_chmod -- Establecer permisos en un archivo via FTP

Descripción

int **ftp_chmod** (resource secuencia_ftp, int modo, string nombre_archivo)

Establece los permisos sobre el archivo remoto especificado por *nombre_archivo* a *modo*, dado como un valor *octal*.

Devuelve *modo* en caso de tener éxito, o **FALSE** de lo contrario.

Ejemplo 1. Ejemplo de ftp_chmod()

```
<?php
$archivo = 'public_html/index.php';

// configurar una conexion basica
$id_con = ftp_connect($servidor_ftp);

// iniciar sesion con nombre de usuario y contrasenya
$resultado_login = ftp_login($id_con, $nombre_usuario_ftp, $contrasenya_ftp);

// intentar chmod en $archivo con el valor 644
if (ftp_chmod($id_con, 0644, $archivo) !== false) {
 echo "se efectu&oacute; chmod sobre $archivo satisfactoriamente con el valor 644\n";
} else {
 echo "no se pudo realizar chmod sobre $archivo\n";
}

// cerrar la conexion
ftp_close($id_con);
?>
```

Devuelve los nuevos permisos del archivo en caso de éxito, o **FALSE** si ocurre un fallo.

Vea también **chmod()**.

ftp_close

(PHP 4 >= 4.2.0, PHP 5)

ftp_close -- Cierra una conexión FTP

Descripción

bool **ftp_close** (resource secuencia_ftp)

ftp_close() cierra *secuencia_ftp* y libera el **resource**. Después de llamar esta función, no puede continuar usando la conexión FTP y debe crear una nueva con **ftp_connect()**.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplo 1. Ejemplo de ftp_close()

```
<?php

// establecer una conexion basica
$id_con = ftp_connect($servidor_ftp);

// iniciar sesion con nombre de usuario y contrasenya
$resultado_login = ftp_login($id_con, $nombre_usuario, $contrasenya_ftp);

// imprimir el directorio actual
echo ftp_pwd($id_con); // /

// cerrar esta conexion
ftp_close($id_con);
?>
```

Vea también **ftp_connect()**

ftp_connect

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

ftp_connect -- Establece una conexión FTP

Descripción

int **ftp_connect** (string host [, int port])

Si tiene éxito, devuelve un flujo FTP. En caso de error, devuelve **FALSE**.

ftp_connect() establece una conexión FTP al *host* especificado. El parámetro *port* especifica un puerto alternativo al que conectar. Si se omite o es cero, se usa el puerto FTP por defecto, 21.

ftp_delete

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

ftp_delete -- Elimina un archivo en el servidor FTP

Descripción

bool **ftp_delete** (resource secuencia_ftp, string ruta)

ftp_delete() elimina el archivo especificado por *ruta* desde el servidor FTP.

Ejemplo 1. Ejemplo de ftp_delete()

```
<?php
$archivo = 'public_html/antiguo.txt';

// establecer conexion basica
$id_con = ftp_connect($servidor_ftp);

// iniciar sesion con nombre de usuario y contrasenya
$resultado_login = ftp_login($id_con, $ftp_nombre_usuario, $ftp_contrasenya);

// intento de eliminar $archivo
if (ftp_delete($id_con, $archivo)) {
 echo "$archivo se elimin&oacute; satisfactoriamente\n";
} else {
 echo "no se pudo eliminar $archivo\n";
}

// cerrar la conexion
ftp_close($id_con);
?>
```

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

ftp_exec

(PHP 4 >= 4.0.3, PHP 5)

ftp_exec -- Solicita la ejecución de un programa en el servidor FTP

Descripción

bool **ftp_exec** (resource secuencia_ftp, string comando)

Envía una petición SITE EXEC *comando* al servidor FTP. Devuelve **TRUE** si el comando tuvo éxito (el servidor envía el código de respuesta: *200*); o **FALSE** de lo contrario.

Ejemplo 1. Ejemplo de ftp_exec()

```
<?php

$comando = 'ls -al >archivos.txt';

$id_con = ftp_connect($servidor_ftp);

$resultado_login = ftp_login($id_con, $nombre_usuario_ftp, $contrasenya_usuario_ftp);

if (ftp_exec($id_con, $comando)) {
    echo "$comando fue ejecutado satisfactoriamente<br />\n";
} else {
    echo 'no se pudo ejecutar ' . $comando;
}

?>
```

Vea también **ftp_raw()**.

ftp_fget

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

ftp_fget -- Descarga un archivo desde el servidor FTP y lo guarda en un archivo abierto

Descripción

bool **ftp_fget** (resource secuencia_ftp, resource gestor, string archivo_remoto, int modo [, int pos_reanudar])

ftp_fget() recupera *archivo_remoto* desde el servidor FTP, y lo escribe en el apuntador de archivo dado, *gestor*. El *modo* de transferencia especificado debe ser un valor entre **`FTP_ASCII`** y **`FTP_BINARY`**.

Ejemplo 1. Ejemplo de ftp_fget()

```
<?php

// abrir algun archivo para lectura
$archivo = 'algun_archivo.txt';
$da = fopen($archivo, 'w');

// establecer la conexion basica
$id_con = ftp_connect($servidor_ftp);

// iniciar sesion con nombre de usuario y contrasenya
$resultado_login = ftp_login($id_con, $ftp_nombre_usuario, $ftp_contrasenya);

// intento por descargar $archivo
if (ftp_fget($id_con, $da, $archivo, FTP_ASCII, 1)) {
 echo "Se ha escrito satisfactoriamente sobre $archivo\n";
} else {
 echo "Hubo un problema con $archivo\n";
}

// cerrar la conexion y el gestor de archivo
ftp_close($id_con);
fclose($da);
?>
```

Nota: El parámetro *pos_reanudar* fue agregado en PHP 4.3.0.

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Vea también **ftp_get()**, **ftp_nb_get()** y **ftp_nb_fget()**.

ftp_fput

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

ftp_fput -- Carga un archivo abierto al servidor FTP

Descripción

bool **ftp_fput** (resource secuencia_ftp, string archivo_remoto, resource gestor, int modo [, int pos_inicio])

ftp_fput() carga los datos desde el apuntador de archivo *gestor* hasta que se llega al final del archivo. Los resultados son almacenados en *archivo_remoto* en el servidor FTP. El *modo* de transferencia especificado debe ser un valor entre **`FTP_ASCII`** y **`FTP_BINARY`**.

Ejemplo 1. Ejemplo de ftp_fput()

```php
<?php

// abrir algun archivo para lectura
$archivo = 'somefile.txt';
$da = fopen($archivo, 'r');

// configurar la conexion basica
$id_con = ftp_connect($servidor_ftp);

// iniciar sesion con nombre de usuario y contrasenya
$resultado_login = ftp_login($id_con, $ftp_nombre_usuario, $ftp_contrasenya);

// trata de cargar $archivo
if (ftp_fput($id_con, $archivo, $da, FTP_ASCII)) {
    echo "Se ha cargado $archivo satisfactoriamente\n";
} else {
    echo "Hubo un problema durante la carga de $archivo\n";
}

// cerrar la conexion y el gestor de archivo
ftp_close($id_con);
fclose($da);

?>
```

Nota: El parámetro *pos_inicio* fue añadido en PHP 4.3.0.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Vea también **ftp_put()**, **ftp_nb_fput()**, y **ftp_nb_put()**.

ftp_get_option

(PHP 4 >= 4.2.0, PHP 5)

ftp_get_option -- Recupera varios comportamientos de tiempo de ejecución de la secuencia FTP actual

Descripción

mixed **ftp_get_option** (resource secuencia_ftp, int opcion)

Devuelve el valor en caso de éxito o **FALSE** si la *opcion* dada no es soportada. En el último caso, un mensaje de advertencia es producido también.

Esta función devuelve el valor para la *opcion* solicitada, de la *secuencia_ftp* especificada. Actualmente, las siguientes opciones son soportadas:

Tabla 1. Opciones FTP de tiempo de ejecución soportadas

FTP_TIMEOUT_SEC	Devuelve el tiempo máximo de espera actual usado para las operaciones de red.

Ejemplo 1. Ejemplo de ftp_get_option()

```php
<?php
// Obtiene el tiempo de espera de la secuencia FTP dada
$timeout = ftp_get_option($id_con, FTP_TIMEOUT_SEC);
?>
```

Vea también **ftp_set_option()**.

ftp_get

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

ftp_get -- Descarga un archivo desde el servidor FTP

Descripción

bool **ftp_get** (resource secuencia_ftp, string archivo_local, string archivo_remoto, int modo [, int pos_reanudar])

ftp_get() recupera *archivo_remoto* desde el servidor FTP, y lo guarda localmente en *archivo_local*. El *modo* de transferencia especificado debe ser un valor entre **`FTP_ASCII`** o **`FTP_BINARY`**.

> **Nota:** El parámetro *pos_reanudar* fue añadido en PHP 4.3.0.

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Ejemplo 1. Ejemplo de ftp_get()

```
<?php

// definir algunas variables
$archivo_local = 'local.zip';
$archivo_servidor = 'servidor.zip';

// configurar conexion basica
$id_con = ftp_connect($servidor_ftp);

// iniciar sesion con nombre de usuario y contrasenya
$resultado_login = ftp_login($id_con, $ftp_nombre_usuario, $ftp_contrasenya);

// intentar la descarga de $archivo_servidor y guardarlo en $archivo_local
if (ftp_get($id_con, $archivo_local, $archivo_servidor, FTP_BINARY)) {
    echo "Se ha guardado satisfactoriamente en $archivo_local\n";
} else {
    echo "Ha ocurrido un problema\n";
}

// cerrar la conexion
ftp_close($id_con);

?>
```

Vea también **ftp_fget()**, **ftp_nb_get()** y **ftp_nb_fget()**.

ftp_login

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

ftp_login -- Comienza la sesion en una conexión FTP

Descripción

int **ftp_login** (int ftp_stream, string username, string password)

Si tiene éxito, devuelve **`TRUE`**. En caso de error, devuelve **`FALSE`**.

Inicia una sesion (envía identificador de usuario y contraseña) en el flujo FTP especificado.

ftp_mdtm

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

ftp_mdtm -- Devuelve la última hora de modificación del archivo dado

Descripción

int **ftp_mdtm** (resource secuencia_ftp, string archivo_remoto)

ftp_mdtm() revisa la última fecha de modificación de un archivo, y la devuelve como una marca de tiempo Unix. Si ocurre un error, o el archivo no existe, se devuelve -1.

Devuelve una marca de tiempo Unix de tener éxito, o -1 en caso de fallo.

Ejemplo 1. Ejemplo de ftp_mdtm()

```
<?php

$archivo = 'algun_archivo.txt';

// configurar la conexion basica
$id_con = ftp_connect($servidor_ftp);

// iniciar sesion con nombre de usuario y contrasenya
$resultado_login = ftp_login($id_con, $ftp_nombre_usuario, $ftp_contrasenya);

// obtener la ultima fecha/hora de modificacion
$buf = ftp_mdtm($id_con, $archivo);

if ($buf != -1) {
    // algun_archivo.txt fue modificado por ultima vez en: March 26 2003 14:16:41.
    echo "$archivo fue modificado por &uacute;ltima vez en: " . date("F d Y H:i:s.", $b
} else {
    echo "No se pudo obtener mdtime";
}

// cerrar la conexion
ftp_close($id_con);

?>
```

Nota: ¡No todos los servidores soportan esta característica!

Nota: ftp_mdtm() no funciona con directorios.

ftp_mkdir

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

ftp_mkdir -- Crea un directorio

Descripción

string **ftp_mkdir** (int ftp_stream, string directory)

Si tiene éxito, devuelve el nombre del directorio recién creado. En caso de error, devuelve **FALSE**.

Crea el directorio especificado por el parámetro *directory*.

ftp_nb_continue

(PHP 4 >= 4.3.0, PHP 5)

ftp_nb_continue -- Continúa recuperando/enviando un archivo (modo no-bloqueo)

Descripción

int **ftp_nb_continue** (resource secuencia_ftp)

Continúa la recuperación/envío de un archivo en modo de no-bloqueo.

Ejemplo 1. Ejemplo de ftp_nb_continue()

```
<?php

// Iniciar la descarga
$ret = ftp_nb_get($mi_conexion, "prueba", "LEAME", FTP_BINARY);
while ($ret == FTP_MOREDATA) {

   // Continuar con la descarga...
   $ret = ftp_nb_continue($mi_conexion);
}
if ($ret != FTP_FINISHED) {
   echo "Hubo un error en la descarga del archivo...";
   exit(1);
}
?>
```

Devuelve **FTP_FAILED** o **FTP_FINISHED** o **FTP_MOREDATA**.

ftp_nb_fget

(PHP 4 >= 4.3.0, PHP 5)

ftp_nb_fget -- Recupera un archivo desde el servidor FTP y lo escribe sobre un archivo abierto (modo no-bloqueo)

Descripción

int **ftp_nb_fget** (resource secuencia_ftp, resource gestor, string archivo_remoto, int modo [, int pos_continuacion])

ftp_nb_fget() recupera *archivo_remoto* desde el servidor FTP, y lo escribe sobre el apuntador de archivo dado, *gestor*. El *modo* de transferencia dado debe ser **FTP_ASCII** o **FTP_BINARY**. La diferencia entre ésta función y **ftp_fget()** es que ésta función recupera el archivo asincrónicamente, de modo que su programa puede realizar otras operaciones mientras el archivo está siendo descargado.

Ejemplo 1. Ejemplo de ftp_nb_fget()

```php
<?php

// abrir un archivo para lectura
$archivo = 'index.php';
$da = fopen($archivo, 'w');

$id_con = ftp_connect($servidor_ftp);

$resultado_login = ftp_login($id_con, $nombre_usuario_ftp, $contrasenya_ftp);

// Iniciar la descarga
$ret = ftp_nb_fget($id_con, $da, $archivo, FTP_BINARY);
while ($ret == FTP_MOREDATA) {

   // Haga lo que desee
   echo ".";

   // Continuar la descarga...
   $ret = ftp_nb_continue($id_con);
}
if ($ret != FTP_FINISHED) {
   echo "Hubo un error en la descarga del archivo...";
   exit(1);
}

// cerrar el apuntador de archivo
fclose($da);
?>
```

Devuelve **`FTP_FAILED`**, **`FTP_FINISHED`**, o **`FTP_MOREDATA`**.

Vea también **ftp_nb_get()**, **ftp_nb_continue()**, **ftp_fget()**, y **ftp_get()**.

ftp_nb_fput

(PHP 4 >= 4.3.0, PHP 5)

ftp_nb_fput -- Almacena un archivo desde un archivo abierto en el servidor FTP (modo no-bloqueo)

Descripción

int **ftp_nb_fput** (resource secuencia_ftp, string archivo_remoto, resource gestor, int modo [, int pos_comienzo])

ftp_nb_fput() carga los datos desde el apuntador de archivo *gestor* hasta que alcanza el fin de archivo. Los resultados son almacenados en *archivo_remoto* en el servidor FTP. El *modo* de transferencia especificado debe ser **`FTP_ASCII`** o **`FTP_BINARY`**. La diferencia entre ésta función y **ftp_fput()** es que ésta función carga el archivo asincrónicamente, de modo que su programa puede realizar otras operaciones mientras que el archivo está siendo cargado.

Ejemplo 1. Ejemplo de ftp_nb_fput()

```php
<?php

$archivo = 'index.php';

$da = fopen($archivo, 'r');

$id_con = ftp_connect($servidor_ftp);

$resultado_login = ftp_login($id_con, $nombre_usuario_ftp, $contrasenya_ftp);

// Iniciar la carga
$ret = ftp_nb_fput($id_con, $arcivo, $da, FTP_BINARY);
while ($ret == FTP_MOREDATA) {

   // Haga lo que desee
   echo ".";

   // Continuar la carga...
   $ret = ftp_nb_continue($id_con);
}
if ($ret != FTP_FINISHED) {
   echo "Hubo un error en la carga del archivo...";
   exit(1);
}

fclose($da);
?>
```

Devuelve **FTP_FAILED**, **FTP_FINISHED**, o **FTP_MOREDATA**.

Vea también **ftp_nb_put()**, **ftp_nb_continue()**, **ftp_put()** y **ftp_fput()**.

ftp_nb_get

(PHP 4 >= 4.3.0, PHP 5)

ftp_nb_get -- Recupera un archivo desde el servidor FTP y lo escribe sobre un archivo local (modo no-bloqueo)

Descripción

int **ftp_nb_get** (resource secuencia_ftp, string archivo_local, string archivo_remoto, int modo [, int pos_continuacion])

ftp_nb_get() recupera el *archivo_remoto* desde el servidor FTP, y lo guarda localmente en *archivo_local*. El *modo* de transferencia especificado debe ser **FTP_ASCII** o **FTP_BINARY**.La diferencia entre ésta función y **ftp_get()** es que ésta función recupera el archivo asincrónicamente, de modo que su programa puede realizar otras operaciones mientras el archivo está siendo descargado.

Devuelve **FTP_FAILED**, **FTP_FINISHED**, o **FTP_MOREDATA**.

Ejemplo 1. Ejemplo de ftp_nb_get()

```
<?php

// Iniciar la descarga
$ret = ftp_nb_get($mi_conexion, "test", "README", FTP_BINARY);
while ($ret == FTP_MOREDATA) {

   // Haga lo que quiera
   echo ".";

   // Continuar la descarga...
   $ret = ftp_nb_continue($mi_conexion);
}
if ($ret != FTP_FINISHED) {
   echo "Hubo un error descargando el archivo...";
   exit(1);
}
?>
```

Ejemplo 2. Reanudando una descarga con ftp_nb_get()

```
<?php

// Iniciar
$ret = ftp_nb_get($mi_conexion, "test", "README", FTP_BINARY,
                  filesize("test"));

// O: $ret = ftp_nb_get($mi_conexion, "test", "README",
//                      FTP_BINARY, FTP_AUTORESUME);
while ($ret == FTP_MOREDATA) {

   // Haga lo que quiera
   echo ".";

   // Continuar descargando...
   $ret = ftp_nb_continue($mi_conexion);
}
if ($ret != FTP_FINISHED) {
   echo "Hubo un error descargando el archivo...";
   exit(1);
}
?>
```

Ejemplo 3. Reanudar una descarga en la posición 100 sobre un archivo nuevo con ftp_nb_get ()

```
<?php

// Deshabilitar Autoseek
ftp_set_option($mi_conexion, FTP_AUTOSEEK, FALSE);

// Iniciar
$ret = ftp_nb_get($mi_conexion, "archivo_nuevo", "README", FTP_BINARY, 100);
while ($ret == FTP_MOREDATA) {

   /* ... */

   // Continuar descarga...
   $ret = ftp_nb_continue($mi_conexion);
}
?>
```

En el ejemplo anterior, "archivo_nuevo" es 100 bytes más pequeña que "README" en el servidor FTP, ya que comenzamos la lectura en la posición 100. Si no hemos deshabilitado **FTP_AUTOSEEK**, los primeros 100 bytes de "archivo_nuevo" serán *'\0'*.

Vea también **ftp_nb_fget()**, **ftp_nb_continue()**, **ftp_get()**, y **ftp_fget()**.

ftp_nb_put

(PHP 4 >= 4.3.0, PHP 5)

ftp_nb_put -- Almacena un archivo en el servidor FTP (modo no-bloqueo)

Descripción

int **ftp_nb_put** (resource secuencia_ftp, string archivo_remoto, string archivo_local, int modo [, int pos_comienzo])

ftp_nb_put() almacena *archivo_local* en el servidor FTP, como *archivo_remoto*. El *modo* de transferencia especificado debe ser **FTP_ASCII** o **FTP_BINARY**. La diferencia entre ésta función y **ftp_put()** es que ésta carga el archivo asincrónicamente, de modo que su programa puede realizar otras operaciones mientras el archivo está siendo cargado.

Devuelve **FTP_FAILED**, **FTP_FINISHED**, o **FTP_MOREDATA**.

Ejemplo 1. Ejemplo de ftp_nb_put()

```
<?php

// Iniciar la carga
$ret = ftp_nb_put($mi_conexion, "test.remoto", "test.local", FTP_BINARY);
while ($ret == FTP_MOREDATA) {

   // Haga lo que quiera
   echo ".";

   // Continuar la carga...
   $ret = ftp_nb_continue($mi_conexion);
}
if ($ret != FTP_FINISHED) {
   echo "Hubo un error cargando el archivo...";
   exit(1);
}
?>
```

Ejemplo 2. Reanudando una carga con ftp_nb_put()

```
<?php

// Iniciar
$ret = ftp_nb_put($mi_conexion, "test.remoto", "test.local",
                  FTP_BINARY, ftp_size("test.remoto"));

// O: $ret = ftp_nb_put($mi_conexion, "test.remoto", "test.local",
//                      FTP_BINARY, FTP_AUTORESUME);

while ($ret == FTP_MOREDATA) {

   // Haga lo que quiera
   echo ".";

   // Continuar la carga...
   $ret = ftp_nb_continue($mi_conexion);
}
if ($ret != FTP_FINISHED) {
   echo "Hubo un error cargando el archivo...";
   exit(1);
}
?>
```

Vea también **ftp_nb_fput()**, **ftp_nb_continue()**, **ftp_put()**, y **ftp_fput()**.

ftp_nlist

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

ftp_nlist -- Devuelve una lista de archivos en el directorio dado

Descripción

array **ftp_nlist** (resource secuencia_ftp, string directorio)

Devuelve una matriz con nombres de archivo del directorio especificado de tener éxito, o **FALSE** si se presenta un fallo.

Ejemplo 1. Ejemplo de ftp_nlist()

```
<?php

// configurar la conexion basica
$id_con = ftp_connect($servidor_ftp);

// iniciar sesion con nombre de usuario y contrasenya
$resultado_login = ftp_login($id_con, $ftp_nombre_usuario, $ftp_contrasenya);

// obtener los contenidos del directorio actual
$contenidos = ftp_nlist($id_con, ".");

// imprimir $contenidos
var_dump($contenidos);

?>
```

El ejemplo anterior imprimirá una salida similar a:

```
array(3) {
  [0]=>
  string(11) "public_html"
  [1]=>
  string(10) "public_ftp"
  [2]=>
  string(3) "www"
```

Vea también **ftp_rawlist()**.

ftp_pasv

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

ftp_pasv -- Activa o desactiva el modo pasivo.

Descripción

int **ftp_pasv** (int ftp_stream, int pasv)

Si tiene éxito, devuelve **TRUE**. En caso de error, devuelve **FALSE**.

ftp_pasv() activa el modo pasivo si el parámetro *pasv* es **TRUE** (desactiva el modo pasivo si *pasv* es **FALSE**.) En modo pasivo, las conexiones de datos son iniciadas por el cliente, en lugar de ser iniciadas por el servidor.

ftp_put

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

ftp_put -- Carga un archivo al servidor FTP

Descripción

bool **ftp_put** (resource secuencia_ftp, string archivo_remoto, string archivo_local, int modo [, int pos_inicio])

ftp_put() almacena *archivo_local* en el servidor FTP, como *archivo_remoto*. El *modo* de transferencia especificado debe ser **`FTP_ASCII`** o **`FTP_BINARY`**.

> **Nota:** El parámetro *pos_inicio* fue agregado en PHP 4.3.0.

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Ejemplo 1. Ejemplo de ftp_put()

```
<?php
$archivo = 'algun-archivo.txt';
$archivo_remoto = 'leame.txt';

// configurar la conexion basica
$id_con = ftp_connect($servidor_ftp);

// iniciar sesion con nombre de usuario y contrasenya
$resultado_login = ftp_login($id_con, $ftp_nombre_usuario, $ftp_contrasenya);

// cargar un archivo
if (ftp_put($id_con, $archivo_remoto, $archivo, FTP_ASCII)) {
 echo "se ha cargado $archivo satisfactoriamente\n";
} else {
 echo "Hubo un problema durante la transferencia de $archivo\n";
}

// cerrar la conexion
ftp_close($id_con);
?>
```

Vea también **ftp_fput()**, **ftp_nb_fput()**, y **ftp_nb_put()**.

ftp_pwd

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

ftp_pwd -- Devuelve el nombre del directorio actual

Descripción

int **ftp_pwd** (int ftp_stream)

Devuelve el directorio actual, o **`FALSE`** en caso de error.

ftp_quit

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

ftp_quit -- Cierra una conexión FTP

Descripción

int **ftp_quit** (int ftp_stream)

ftp_connect() cierra el flujo FTP *ftp_stream*.

ftp_raw

(PHP 5)

ftp_raw -- Envía un comando arbitrario a un servidor FTP

Descripción

array **ftp_raw** (resource secuencia_ftp, string comando)

Envía un *comando* arbitrario al servidor FTP. Devuelve la respuesta del servidor como una matriz de cadenas. No se realiza procesamiento alguno con la cadena de respuesta, ni determina **ftp_raw()** si el comando tuvo éxito.

Ejemplo 1. Uso de ftp_raw() para iniciar una sesión con un servidor FTP manualmente.

```
<?php
$desc = ftp_connect("ftp.example.com");

/* Esto es igual a:
   ftp_login($desc, "joeblow", "secret"); */
ftp_raw($desc, "USER joeblow");
ftp_raw($desc, "PASS secret");
?>
```

Vea también: **ftp_exec()**

ftp_rawlist

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

ftp_rawlist -- Devuelve una lista detallada de archivos en el directorio dado

Descripción

array **ftp_rawlist** (resource secuencia_ftp, string directorio [, bool recursivo])

ftp_rawlist() ejecuta el comando FTP LIST, y devuelve el resultado en forma de matriz. Cada elemento de la matriz corresponde a una línea de texto. La salida no es procesada en forma alguna. El identificador de tipo de sistema devuelto por **ftp_systype()** puede ser usado para determinar la forma en que los resultados deberían ser interpretados.

El parámetro opcional *recursivo* se encuentra disponible a partir de PHP 4.3.0.

Ejemplo 1. Ejemplo de ftp_rawlist()

```
<?php

// establecer una conexion basica
$id_con = ftp_connect($servidor_ftp);

// iniciar sesion con nombre de usuario y contrasenya
$resultado_login = ftp_login($id_con, $ftp_nombre_usuario, $ftp_contrasenya);

// obtener la lista de archivos de /
$bufer = ftp_rawlist($id_con, '/');

// cerrar la conexion
ftp_close($id_con);

// imprimir el contenido del bufer
var_dump($bufer);
?>
```

El ejemplo anterior producirá una salida similar a: Ã, Ã,

```
array(3) {
  [0]=>
  string(65) "drwxr-x---   3 vincent  vincent      4096 Jul 12 12:16 public_ftp"
  [1]=>
  string(66) "drwxr-x---  15 vincent  vincent      4096 Nov  3 21:31 public_html"
  [2]=>
  string(73) "lrwxrwxrwx   1 vincent  vincent        11 Jul 12 12:16 www -> public_html
}
```

Vea también **ftp_nlist()**.

ftp_rename

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

ftp_rename -- Renombra un archivo en el servidor FTP

Descripción

bool **ftp_rename** (resource secuencia_ftp, string original, string destino)

ftp_rename() renombra el archivo o directorio llamado actualmente *original* al nuevo nombre *destino*, usando la secuencia FTP *secuencia_ftp*.

Ejemplo 1. Ejemplo de ftp_rename()

```
<?php
$archivo_viejo = 'un_archivo.txt.bak';
$archivo_nuevo = 'un_archivo.txt';

// establecer la conexion basica
$id_con = ftp_connect($ftp_server);

// iniciar sesion con nombre de usuario y contrasenya
$resultado_login = ftp_login($id_con, $ftp_nombre_usuario, $ftp_contrasenya);

// tratar de renombrar $archivo_viejo a $archivo_nuevo
if (ftp_rename($id_con, $archivo_viejo, $archivo_nuevo)) {
 echo "se ha renombrado $archivo_viejo a $archivo_nuevo con &eacute;xito\n";
} else {
 echo "Hubo un problema al renombrar $archivo_viejo a $archivo_nuevo\n";
}

// cerrar la conexion
ftp_close($id_con);
?>
```

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

ftp_rmdir

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

ftp_rmdir -- Borra un directorio

Descripción

int **ftp_rmdir** (int ftp_stream, string directory)

Si tiene éxito, devuelve **TRUE**. En caso de error, devuelve **FALSE**.

Borra el directorio especificado por el parámetro *directory*.

ftp_set_option

(PHP 4 >= 4.2.0, PHP 5)

ftp_set_option -- Establecer varias opciones FTP de tiempo de ejecución

Descripción

bool **ftp_set_option** (resource secuencia_ftp, int opcion, mixed valor)

Devuelve **TRUE** si la opción pudo definirse; **FALSE** de lo contrario. Un mensaje de advertencia será producido si la *opcion* no es soportada, o el *valor* pasado no coincide con el valor esperado para la *opcion* dada.

Esta función controla varias opciones de tiempo de ejecución para la secuencia FTP especificada. El parámetro *valor* depende de cuál parámetro *opcion* es seleccionado para ser alterado. En la actualidad, las siguientes opciones son soportadas:

Tabla 1. Opciones FTP de tiempo de ejecución soportadas

FTP_TIMEOUT_SEC	Modifica el tiempo máximo de espera en segundos usado para todas las funciones de red. *valor* debe ser un entero mayor a 0. El tiempo de espera predeterminado es 90 segundos.
FTP_AUTOSEEK	Cuando se encuentra habilitado, las peticiones GET o PUT con un parámetro *pos_continuacion* o *pos_comienzo* buscarán primero la posición solicitada dentro del archivo. Este parámetro está habilitado por defecto.

Ejemplo 1. Ejemplo de ftp_set_option()

```
<?php
// Establecer el tiempo de espera de red a 10 segundos
ftp_set_option($id_con, FTP_TIMEOUT_SEC, 10);
?>
```

Vea también **ftp_get_option()**.

ftp_site

(PHP 3>= 3.0.15, PHP 4 , PHP 5)

ftp_site -- Envía un comando SITE al servidor

Descripción

bool **ftp_site** (resource secuencia_ftp, string cmd)

ftp_site() envía el comando especificado por *cmd* al servidor FTP. Los comandos SITE no se encuentran estandarizados, y varían de un servidor a otro. Son útiles para la gestión de cosas como permisos de archivos y membresías de grupos.

Ejemplo 1. Envío de un comando SITE a un servidor ftp

```
<?php
/* Conectarse con el servidor FTP */
$con = ftp_connect('ftp.example.com');
if (!$con) die('No fue posible conectarse con ftp.example.com');

/* Iniciar sesion como "usuario" con la contrasenya "pass" */
if (!ftp_login($con, 'usuario', 'pass')) die('Error iniciando sesion en ftp.example.com

/* Aplicar el comando "SITE CHMOD 0600 /home/usuario/archivoprivado"
 * al servidor ftp */
if (ftp_site($con, 'CHMOD 0600 /home/usuario/archivoprivado')) {
   echo "Comando ejecutado satisfactoriamente.\n";
} else {
   die('El comando ha fallado.');
}
?>
```

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Vea también: **ftp_raw()**

ftp_size

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

ftp_size -- Devuelve el tamaño del archivo dado

Descripción

int **ftp_size** (resource secuencia_ftp, string archivo_remoto)

ftp_size() devuelve el tamaño de un *archivo_remoto* en bytes. Si ocurre un error, o si el archivo dado no existo, o es un directorio, el valor -1 es devuelto. No todos los servidores soportan esta característica.

Devuelve el tamaño del archivo de tener éxito, o -1 en caso de fallo.

Ejemplo 1. Ejemplo de ftp_size()

```
<?php

$archivo = 'un_archivo.txt';

// establecer la conexion basica
$id_con = ftp_connect($servidor_ftp);

// iniciar sesion con nombre de usuario y contrasenya
$resultado_login = ftp_login($id_con, $ftp_nombre_usuario, $ftp_contrasenya);

// obtener el tamanyo de $archivo
$res = ftp_size($id_con, $archivo);

if ($res != -1) {
    echo "el tama&ntilde;o de $file es $res bytes";
} else {
    echo "no pudo obtenerse el tama&ntilde;o";
}

// cerrar la conexion
ftp_close($id_con);

?>
```

Vea también **ftp_rawlist()**.

ftp_ssl_connect

(PHP 4 >= 4.3.0, PHP 5)

ftp_ssl_connect -- Abre una conexión segura SSL-FTP

Descripción

resource **ftp_ssl_connect** (string host [, int puerto [, int tiempo_espera]])

Devuelve una secuencia SSL-FTP en caso de éxito, o **FALSE** si ocurre un fallo.

ftp_ssl_connect() abre una conexión SSL-FTP con el *host* especificado. El parámetro *puerto* especifica un puerto alternativo para realizar la conexión. Si es omitido o definido como cero, entonces será usado el puerto FTP predeterminado, 21.

El parámetro *tiempo_espera* especifica el tiempo máximo de espera para todas las operaciones de red subsecuentes. Si se omite, el valor predeterminado es 90 segundos. El tiempo de espera puede ser modificado y consultado en cualquier momento con **ftp_set_option()** y **ftp_get_option()**.

Ejemplo 1. Ejemplo de ftp_ssl_connect()

```php
<?php

// establecer una conexion ssl basica
$id_con = ftp_ssl_connect($servidor_ftp);

// iniciar sesion con nombre de usuario y contrasenya
$resultado_login = ftp_login($id_con, $nombre_usuario_ftp, $contrasenya_ftp);

echo ftp_pwd($id_con); // /

// cerrar la conexion ssl
ftp_close($id_con);
?>
```

> **Porqué esta función puede no existir: ftp_ssl_connect()** se encuentra disponible únicamente si el soporte para OpenSSL está habilitado en su versión de PHP. Si no está definida y ha compilado el soporte para FTP, entonces ésta es la causa. Para Windows, debe compilar sus propios binarios de PHP para contar con soporte para esta función.

Vea también **ftp_connect()**.

ftp_systype

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

ftp_systype -- Devuelve el identificador de tipo de sistema del servidor FTP remoto.

Descripción

int **ftp_systype** (int ftp_stream)

Devuelve el tipo de sistema remoto, o **FALSE** en caso de error.

XL. Funciones de Gestión de Funciones

Introducción

Estas funciones gestionan varias operaciones involucradas con el trabajo con funciones.

Requirimientos

No se necesitan bibliotecas externas para construir esta extensión

---

Instalación

No se necesita ninguna instalación para usar estas funciones, son parte del núcleo de PHP.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Constantes predefinidas

Esta extensión no tiene ninguna constante definida.

Tabla de contenidos

call_user_func_array

(PHP 4 >= 4.0.4, PHP 5)

call_user_func_array -- Call a user function given with an array of parameters

Description

mixed **call_user_func_array** (callback function, array param_arr)

Call a user defined function given by *function*, with the parameters in *param_arr*. For example:

Ejemplo 1. call_user_func_array() example

```
<?php
function debug($var, $val)
{
    echo "***DEBUGGING\nVARIABLE: $var\nVALUE:";
    if (is_array($val) || is_object($val) || is_resource($val)) {
        print_r($val);
    } else {
        echo "\n$val\n";
    }
    echo "***\n";
}

$c = mysql_connect();
$host = $_SERVER["SERVER_NAME"];

call_user_func_array('debug', array("host", $host));
call_user_func_array('debug', array("c", $c));
call_user_func_array('debug', array("_POST", $_POST));
?>
```

See also **call_user_func()**, and information about the callback type

call_user_func

(PHP 3>= 3.0.3, PHP 4 , PHP 5)

call_user_func -- Call a user function given by the first parameter

Description

mixed **call_user_func** (string function_name [, mixed parameter [, mixed ...]])

Call a user defined function given by the *function_name* parameter. Take the following:

```
function barber ($type) {
    print "You wanted a $type haircut, no problem";
}
call_user_func ('barber', "mushroom");
call_user_func ('barber', "shave");
```

create_function

(PHP 4 >= 4.0.1, PHP 5)

create_function -- Create an anonymous (lambda-style) function

Description

string **create_function** (string args, string code)

Creates an anonymous function from the parameters passed, and returns a unique name for it. Usually the *args* will be passed as a single quote delimited string, and this is also recommended for the *code*. The reason for using single quoted strings, is to protect the variable names from parsing, otherwise, if you use double quotes there will be a need to escape the variable names, e.g. *\$avar*.

You can use this function, to (for example) create a function from information gathered at run time:

Ejemplo 1. Creating an anonymous function with create_function()

```
<?php
$newfunc = create_function('$a,$b', 'return "ln($a) + ln($b) = " . log($a * $b);');
echo "New anonymous function: $newfunc\n";
echo $newfunc(2, M_E) . "\n";
// outputs
// New anonymous function: lambda_1
// ln(2) + ln(2.718281828459) = 1.6931471805599
?>
```

Or, perhaps to have general handler function that can apply a set of operations to a list of parameters:

Ejemplo 2. Making a general processing function with create_function()

```
<?php
function process($var1, $var2, $farr)
{
    for ($f=0; $f < count($farr); $f++) {
        echo $farr[$f]($var1, $var2) . "\n";
    }
}

// create a bunch of math functions
$f1 = 'if ($a >=0) {return "b*a^2 = ".$b*sqrt($a);} else {return false;}';
$f2 = "return \"min(b^2+a, a^2,b) = \".min(\$a*\$a+\$b,\$b*\$b+\$a);";
$f3 = 'if ($a > 0 && $b != 0) {return "ln(a)/b = ".log($a)/$b; } else { return false; }
$farr = array(
    create_function('$x,$y', 'return "some trig: ".(sin($x) + $x*cos($y));'),
    create_function('$x,$y', 'return "a hypotenuse: ".sqrt($x*$x + $y*$y);'),
    create_function('$a,$b', $f1),
    create_function('$a,$b', $f2),
    create_function('$a,$b', $f3)
    );

echo "\nUsing the first array of anonymous functions\n";
echo "parameters: 2.3445, M_PI\n";
process(2.3445, M_PI, $farr);

// now make a bunch of string processing functions
$garr = array(
    create_function('$b,$a', 'if (strncmp($a, $b, 3) == 0) return "** \"$a\" '.
    'and \"$b\"\n** Look the same to me! (looking at the first 3 chars)";'),
    create_function('$a,$b', '; return "CRCs: " . crc32($a) . " , ".crc32(b);'),
    create_function('$a,$b', '; return "similar(a,b) = " . similar_text($a, $b, &$p) .
    );
echo "\nUsing the second array of anonymous functions\n";
process("Twas brilling and the slithy toves", "Twas the night", $garr);
?>
```

and when you run the code above, the output will be:

```
Using the first array of anonymous functions
parameters: 2.3445, M_PI
some trig: -1.6291725057799
a hypotenuse: 3.9199852871011
b*a^2 = 4.8103313314525
min(b^2+a, a^2,b) = 8.6382729035898
ln(a/b) = 0.27122299212594

Using the second array of anonymous functions
** "Twas the night" and "Twas brilling and the slithy toves"
** Look the same to me! (looking at the first 3 chars)
CRCs: -725381282 , 1908338681
similar(a,b) = 11(45.833333333333%)
```

But perhaps the most common use for of lambda-style (anonymous) functions is to create callback functions, for example when using **array_walk()** or **usort()**

Ejemplo 3. Using anonymous functions as callback functions

```
<?php
$av = array("the ", "a ", "that ", "this ");
array_walk($av, create_function('&$v,$k', '$v = $v . "mango";'));
print_r($av);
?>
```

outputs:

```
Array
(
  [0] => the mango
  [1] => a mango
  [2] => that mango
  [3] => this mango
)
```

an array of strings ordered from shorter to longer

```
<?php

$sv = array("small", "larger", "a big string", "it is a string thing");
print_r($sv);

?>
```

outputs:

```
Array
(
  [0] => small
  [1] => larger
  [2] => a big string
  [3] => it is a string thing
)
```

sort it from longer to shorter

```
<?php

usort($sv, create_function('$a,$b','return strlen($b) - strlen($a);'));
print_r($sv);

?>
```

outputs:

```
Array
(
  [0] => it is a string thing
  [1] => a big string
  [2] => larger
  [3] => small
)
```

func_get_arg

(PHP 4 , PHP 5)

func_get_arg -- Devuelve un elemento de la lista de argumentos.

Descripción

int **func_get_arg** (int arg_num)

Devuelve el argumento que está en la posición *arg_num* en la lista de argumentos de una función definida por el usuario. Los argumentos de la función se cuentan comenzando por la posición cero. **func_get_arg()** generará un aviso si se llama desde fuera de la definición de la función.

Si *arg_num* es mayor que el número de argumentos pasados realmente, se generará un aviso y **func_get_arg()** devolverá **`FALSE`**.

```
<?php
function foo() {
   $numargs = func_num_args();
   echo "Number of arguments: $numargs<br>\n";
   if ( $numargs >= 2 ) {
      echo "Second argument is: " . func_get_arg( 1 ) . "<br>\n";
   }
}

foo( 1, 2, 3 );
?>
```

func_get_arg() puede utilizarse conjuntamente con **func_num_args()** y **func_get_args()** para permitir a las funciones definidas por el usuario que acepten listas de argumentos de longitud variable.

> **Nota:** Esta función fue añadida en PHP 4.

func_get_args

(PHP 4 , PHP 5)

func_get_args -- Devuelve un array que contiene la lista de argumentos de una función.

Descripción

int **func_get_args** (void)

Devuelve un array en el que cada elemento es el miembro correspondiente de la lista de argumentos de la función definida por el usuario actual. **func_get_args()** generará un aviso si es llamada desde fuera de la definición de la función.

```
<?php
function foo() {
   $numargs = func_num_args();
   echo "Number of arguments: $numargs<br>\n";
   if ( $numargs >= 2 ) {
      echo "Second argument is: " . func_get_arg( 1 ) . "<br>\n";
   }
   $arg_list = func_get_args();
   for ( $i = 0; $i < $numargs; $i++ ) {
      echo "Argument $i is: " . $arg_list[$i] . "<br>\n";
   }
}

foo( 1, 2, 3 );
?>
```

func_get_args() puede utilizarse conjuntamente con **func_num_args()** y **func_get_arg()** para permitir a las funciones definidas por el usuario que acepten listas de argumentos de longitud variable.

> **Nota:** Esta función fue añadida en PHP 4.

func_num_args

(PHP 4 , PHP 5)

func_num_args -- Devuelve el número de argumentos pasados a la función.

Descripción

int **func_num_args** (void)

Devuelve el número de argumentos pasados a la función actual definida por el usuario. **func_num_args()** generará un aviso si es llamada desde fuera de la definición de la función.

```
<?php
function foo() {
   $numargs = func_num_args();
   echo "Number of arguments: $numargs\n";
}

foo( 1, 2, 3 );  // Prints 'Number of arguments: 3'
?>
```

func_num_args() puede utilizarse conjuntamente con **func_get_arg()** y **func_get_args()** para permitir a las funciones definidas por el usuario que acepten listas de argumentos de longitud variable.

Nota: Esta función fue añadida en PHP 4.

function_exists

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

function_exists -- Devuelve **TRUE** si la función dada ha sido definida

Descripción

int **function_exists** (string function_name)

Consulta la lista de funciones definidas buscando *function_name* (nombre de función). Devuelve **TRUE** si encuentra el nombre de función dado, **FALSE** en otro caso.

get_defined_functions

(PHP 4 >= 4.0.4, PHP 5)

get_defined_functions -- Returns an array of all defined functions

Description

array **get_defined_functions** (void)

This function returns an multidimensional array containing a list of all defined functions, both built-in (internal) and user-defined. The internal functions will be accessible via *$arr["internal"]*, and the user defined ones using *$arr["user"]* (see example below).

```
<?php
function myrow($id, $data)
{
    return "<tr><th>$id</th><td>$data</td></tr>\n";
}

$arr = get_defined_functions();

print_r($arr);
?>
```

Will output something along the lines of:

```
Array
(
    [internal] => Array
        (
            [0] => zend_version
            [1] => func_num_args
            [2] => func_get_arg
            [3] => func_get_args
            [4] => strlen
            [5] => strcmp
            [6] => strncmp
            ...
            [750] => bcscale
            [751] => bccomp
        )

    [user] => Array
        (
            [0] => myrow
        )

)
```

See also **function_exists()**, **get_defined_vars()** and **get_defined_constants()**.

register_shutdown_function

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

register_shutdown_function -- Registra una función para su ejecución en el cierre.

Descripción

int **register_shutdown_function** (string func)

Registra la función nombrada en *func* para que se ejecute cuando el script procese su finalización.

Aviso:

Debido a que no se permite ningún tipo de salida en el navegador en esta función, no será capaz de depurarla utilizando sentencias como print o echo.

register_tick_function

(PHP 4 >= 4.0.3, PHP 5)

register_tick_function -- Register a function for execution on each tick

Description

void **register_tick_function** (callback function [, mixed arg [, mixed ...]])

Registers the function named by *func* to be executed when a tick is called. Also, you may pass an array consisting of an object and a method as the *func*.

Ejemplo 1. register_tick_function() example

```
<?php
// using a function as the callback
register_tick_function('my_function', true);

// using an object->method
$object = new my_class();
register_tick_function(array(&$object, 'my_method'), true);
?>
```

See also **declare()** and **unregister_tick_function()**.

unregister_tick_function

(PHP 4 >= 4.0.3, PHP 5)

unregister_tick_function -- De-register a function for execution on each tick

Description

void **unregister_tick_function** (string function_name)

De-registers the function named by *function_name* so it is no longer executed when a tick is called.

XLI. Gettext

Introducción

Las funciones gettext implementan una interfaz de programación de Soporte de Lenguaje Nativo (NLS por sus siglas en Inglés) que puede ser usada para internacionalizar sus aplicaciones PHP. Por favor refiérase a la documentación de gettext en su sistema para una explicación detallada de éstas funciones, o consulte los documentos en http://www.gnu.org/software/gettext/manual/gettext.html.

Requirimientos

Para usar éstas funciones, debe descargar e instalar el paquete GNU gettext desde http://www.gnu.org/software/gettext/gettext.html.

Instalación

Para incluir soporte para GNU gettext en su instalación de PHP, debe agregar la opción *--with-gettext[=DIR]* donde DIR es el directorio de instalación de gettext, por defecto `/usr/local`.

> **Nota para Usuarios de Win32:** Para habilitar este módulo en un entorno Windows, debe copiar `gnu_gettext.dll` desde el directorio DLL del paquete binario de PHP/Win32 a la carpeta SYSTEM32 de su máquina windows. (Ej: `C:\WINNT\SYSTEM32` o `C:\WINDOWS\SYSTEM32`). A partir de PHP 4.2.3 el nombre cambió a `libintl-1.dll`, esto requiere que `iconv.dll` también sea copiado. `libintl-1.dll` no se necesita a partir de PHP 4.3.8, `iconv.dll` no se necesita a partir de PHP 5.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Esta extensión no tiene ninguna constante definida.

Tabla de contenidos

bind_textdomain_codeset

(PHP 4 >= 4.2.0, PHP 5)

bind_textdomain_codeset -- Especificar la codificación de caracteres con la que serán devueltos los mensajes del catálogo DOMINIO

Descripción

string **bind_textdomain_codeset** (string dominio, string juego_caracteres)

Con **bind_textdomain_codeset()**, puede definir la codificación de los mensajes provenientes de *dominio*, devueltos por **gettext()** y funciones similares.

bindtextdomain

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

bindtextdomain -- Establece la ruta para un dominio

Descripción

string **bindtextdomain** (string domain, string directory)

La función **bindtextdomain()** establece la ruta para el dominio.

dcgettext

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

dcgettext -- Sobrescribe el dominio para una búsqueda única

Descripción

string **dcgettext** (string dominio, string mensaje, int categoria)

Esta función le permite sobrescribir el dominio actual para una búsqueda única de mensaje. También le permite especificar una *categoria*.

Vea también **gettext()**.

dcngettext

(PHP 4 >= 4.2.0, PHP 5)

dcngettext -- Versión plural de dcgettext

Descripción

string **dcngettext** (string dominio, string id_mensaje1, string id_mensaje2, int n, int categoria)

Esta función le permite sobrescribir el dominio actual para una consulta única de mensaje en plural. También le permite especificar una *categoria*.

Vea también **ngettext()**.

dgettext

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

dgettext -- Omite el dominio actual

Descripción

string **dgettext** (string domain, string message)

La función **dgettext()** permite omitir el dominio actual para una única busqueda.

dngettext

(PHP 4 >= 4.2.0, PHP 5)

dngettext -- Versión plural de dgettext

Descripción

string **dngettext** (string dominio, string id_mensaje1, string id_mensaje2, int n)

La función **dngettext()** le permite sobrescribir el dominio actual para una consulta única de mensaje en plural.

Vea también **ngettext()**.

gettext

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

gettext -- Realiza una busqueda del mensaje en el dominio actual

Descripción

string **gettext** (string message)

Esta función devuelve la traducción de la cadena si encuentra una en la tabla de traducciones, o el mensaje enviado si no se encuentra ninguna. Puede usar un carácter de subrayado como un alias para esta función.

Ejemplo 1. gettext()-check

```
<?php
// Establece el idioma en alemán
putenv ("LANG=de");

// Especifica la localización de las tablas de traducción
bindtextdomain ("myPHPApp", "./locale");

// Elige un dominio
textdomain ("myPHPApp");

// Imprime un mensaje de prueba
print (gettext ("Welcome to My PHP Application"));
?>
```

ngettext

(PHP 4 >= 4.2.0, PHP 5)

ngettext -- Versión plural de gettext

Descripción

string **ngettext** (string id_mensaje1, string id_mensaje2, int n)

ngettext() devuelve la forma plural correcta del mensaje identificado por *id_mensaje1* e *id_mensaje2* para el conteo *n*. Algunos idiomas tienen más de una forma para mensajes plurales dependiendo del conteo.

Ejemplo 1. Ejemplo de ngettext()

```
<?php

setlocale(LC_ALL, 'cs_CZ');
printf(ngettext("%d ventana", "%d ventanas", 1), 1); // 1 okno
printf(ngettext("%d ventana", "%d ventanas", 2), 2); // 2 okna
printf(ngettext("%d ventana", "%d ventanas", 5), 5); // 5 oken

?>
```

textdomain

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

textdomain -- Establece el dominio actual

Descripción

int **textdomain** ([string library])

Esta función establece el dominio en el que se realizarán las busquedas provocadas por las llamadas a **gettext()**, normalmente el nombre dado a la aplicación. Se devuelve el dominio anterior. Puede llamar a la función sin parámetros para obtener el dominio actual sin necesidad de cambiarlo.

XLII. GMP Functions

Introducción

These functions allow you to work with arbitrary-length integers using the GNU MP library.

These functions have been added in PHP 4.0.4.

> **Nota:** Most GMP functions accept GMP number arguments, defined as *resource* below. However, most of these functions will also accept numeric and string arguments, given that it is possible to convert the latter to a number. Also, if there is a faster function that can operate on integer arguments, it would be used instead of the slower function when the supplied arguments are integers. This is done transparently, so the bottom line is that you can use integers in every function that expects GMP number. See also the **gmp_init()** function.

Aviso

If you want to explicitly specify a large integer, specify it as a string. If you don't do that, PHP will interpret the integer-literal first, possibly resulting in loss of precision, even before *GMP* comes into play.

Nota: This extension is available on Windows platforms since PHP 5.1.0.

Requirimientos

You can download the GMP library from http://www.swox.com/gmp/. This site also has the GMP manual available.

You will need GMP version 2 or better to use these functions. Some functions may require more recent version of the GMP library.

Instalación

In order to have these functions available, you must compile PHP with GMP support by using the *--with-gmp* option.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

`GMP_ROUND_ZERO` (**integer**)

`GMP_ROUND_PLUSINF` (**integer**)

`GMP_ROUND_MINUSINF` (**integer**)

Ejemplos

Ejemplo 1. Factorial function using GMP

```php
<?php
function fact($x)
{
    $return = 1;
    for ($i=2; $i < $x; $i++) {
        $return = gmp_mul($return, $i);
    }
    return $return;
}

echo gmp_strval(fact(1000)) . "\n";
?>
```

This will calculate factorial of 1000 (pretty big number) very fast.

Ver también

More mathematical functions can be found in the sections BCMath Arbitrary Precision Mathematics Functions and Mathematical Functions.

Tabla de contenidos

gmp_sqrtrem -- Square root with remainder
gmp_strval -- Convert GMP number to string
gmp_sub -- Subtract numbers
gmp_xor -- Logical XOR

gmp_abs

(PHP 4 >= 4.0.4, PHP 5)

gmp_abs -- Absolute value

Description

resource **gmp_abs** (resource a)

Returns absolute value of *a*.

gmp_add

(PHP 4 >= 4.0.4, PHP 5)

gmp_add -- Add numbers

Description

resource **gmp_add** (resource a, resource b)

Add two GMP numbers. The result will be GMP number representing the sum of the arguments.

gmp_and

(PHP 4 >= 4.0.4, PHP 5)

gmp_and -- Logical AND

Description

resource **gmp_and** (resource a, resource b)

Calculates logical AND of two GMP numbers.

gmp_clrbit

(PHP 4 >= 4.0.4, PHP 5)

gmp_clrbit -- Clear bit

Description

resource **gmp_clrbit** (resource &a, int index)

Clears (sets to 0) bit *index* in *a*.

gmp_cmp

(PHP 4 >= 4.0.4, PHP 5)

gmp_cmp -- Compare numbers

Description

int **gmp_cmp** (resource a, resource b)

Returns positive value if *a* > *b*, zero if *a* = *b* and negative value if *a* < *b*.

gmp_com

(PHP 4 >= 4.0.4, PHP 5)

gmp_com -- Calculates one's complement

Description

resource **gmp_com** (resource a)

Returns the one's complement of *a*.

Ejemplo 1. gmp_com() example

```
<?php
$com = gmp_com("1234");
echo gmp_strval($com) . "\n";
?>
```

The printout of the above program will be:

```
-1235
```

gmp_div_q

(PHP 4 >= 4.0.4, PHP 5)

gmp_div_q -- Divide numbers

Description

resource **gmp_div_q** (resource a, resource b [, int round])

Divides *a* by *b* and returns the integer result. The result rounding is defined by the *round*, which can

have the following values:

- *GMP_ROUND_ZERO*: The result is truncated towards 0.
- *GMP_ROUND_PLUSINF*: The result is rounded towards +*infinity*.
- *GMP_ROUND_MINUSINF*: The result is rounded towards -*infinity*.

This function can also be called as **gmp_div()**.

See also **gmp_div_r()**, **gmp_div_qr()**

gmp_div_qr

(PHP 4 >= 4.0.4, PHP 5)

gmp_div_qr -- Divide numbers and get quotient and remainder

Description

array **gmp_div_qr** (resource n, resource d [, int round])

The function divides *n* by *d* and returns array, with the first element being *[n/d]* (the integer result of the division) and the second being *(n - [n/d] * d)* (the remainder of the division).

See the **gmp_div_q()** function for description of the *round* argument.

Ejemplo 1. Division of GMP numbers

```
<?php
    $a = gmp_init ("0x41682179fbf5");
    $res = gmp_div_qr ($a, "0xDEFE75");
    printf("Result is: q - %s, r - %s",
            gmp_strval ($res[0]), gmp_strval ($res[1]));
?>
```

See also **gmp_div_q()**, **gmp_div_r()**.

gmp_div_r

(PHP 4 >= 4.0.4, PHP 5)

gmp_div_r -- Remainder of the division of numbers

Description

resource **gmp_div_r** (resource n, resource d [, int round])

Calculates remainder of the integer division of *n* by *d*. The remainder has the sign of the *n* argument, if not zero.

See the **gmp_div_q()** function for description of the *round* argument.

See also **gmp_div_q()**, **gmp_div_qr()**

gmp_div

(PHP 4 >= 4.0.4, PHP 5)

gmp_div -- Divide numbers

Description

resource **gmp_div** (resource a, resource b)

This function is an alias to **gmp_div_q()**.

gmp_divexact

(PHP 4 >= 4.0.4, PHP 5)

gmp_divexact -- Exact division of numbers

Description

resource **gmp_divexact** (resource n, resource d)

Divides *n* by *d*, using fast "exact division" algorithm. This function produces correct results only when it is known in advance that *d* divides *n*.

gmp_fact

(PHP 4 >= 4.0.4, PHP 5)

gmp_fact -- Factorial

Description

resource **gmp_fact** (int a)

Calculates factorial (*a!*) of *a*.

gmp_gcd

(PHP 4 >= 4.0.4, PHP 5)

gmp_gcd -- Calculate GCD

Description

resource **gmp_gcd** (resource a, resource b)

Calculate greatest common divisor of *a* and *b*. The result is always positive even if either of or both

input operands are negative.

gmp_gcdext

(PHP 4 >= 4.0.4, PHP 5)

gmp_gcdext -- Calculate GCD and multipliers

Description

array **gmp_gcdext** (resource a, resource b)

Calculates g, s, and t, such that *a*s + b*t = g = gcd(a,b)*, where gcd is gretest common divisor. Returns array with respective elements g, s and t.

gmp_hamdist

(PHP 4 >= 4.0.4, PHP 5)

gmp_hamdist -- Hamming distance

Description

int **gmp_hamdist** (resource a, resource b)

Returns the hamming distance between *a* and *a*. Both operands should be non-negative.

gmp_init

(PHP 4 >= 4.0.4, PHP 5)

gmp_init -- Create GMP number

Description

resource **gmp_init** (mixed number)

Creates a GMP number from integer or string. String representation can be decimal or hexadecimal. In the latter case, the string should start with *0x*.

Ejemplo 1. Creating GMP number

```
<?php
    $a = gmp_init (123456);
    $b = gmp_init ("0xFFFFDEBACDFEDF7200");
?>
```

> **Nota:** It is not necessary to call this function if you want to use integer or string in place of GMP number in GMP functions, like **gmp_add()**. Function arguments are automatically converted to GMP numbers, if such conversion is possible and needed, using the same rules as **gmp_init()**.

gmp_intval

(PHP 4 >= 4.0.4, PHP 5)

gmp_intval -- Convert GMP number to integer

Description

int **gmp_intval** (resource gmpnumber)

This function allows to convert GMP number to integer.

Aviso
This function returns useful result only if the number actually fits the PHP integer (i.e., signed long type). If you want just to print the GMP number, use **gmp_strval()**.

gmp_invert

(PHP 4 >= 4.0.4, PHP 5)

gmp_invert -- Inverse by modulo

Description

resource **gmp_invert** (resource a, resource b)

Computes the inverse of *a* modulo *b*. Returns **FALSE** if an inverse does not exist.

gmp_jacobi

(PHP 4 >= 4.0.4, PHP 5)

gmp_jacobi -- Jacobi symbol

Description

int **gmp_jacobi** (resource a, resource p)

Computes Jacobi symbol of *a* and *p*. *p* should be odd and must be positive.

gmp_legendre

(PHP 4 >= 4.0.4, PHP 5)

gmp_legendre -- Legendre symbol

Description

int **gmp_legendre** (resource a, resource p)

Compute the Legendre symbol of *a* and *p*. *p* should be odd and must be positive.

gmp_mod

(PHP 4 >= 4.0.4, PHP 5)

gmp_mod -- Modulo operation

Description

resource **gmp_mod** (resource n, resource d)

Calculates *n* modulo *d*. The result is always non-negative, the sign of *d* is ignored.

gmp_mul

(PHP 4 >= 4.0.4, PHP 5)

gmp_mul -- Multiply numbers

Description

resource **gmp_mul** (resource a, resource b)

Multiplies *a* by *b* and returns the result.

gmp_neg

(PHP 4 >= 4.0.4, PHP 5)

gmp_neg -- Negate number

Description

resource **gmp_neg** (resource a)

Returns -*a*.

gmp_or

(PHP 4 >= 4.0.4, PHP 5)

gmp_or -- Logical OR

Description

resource **gmp_or** (resource a, resource b)

Calculates logical inclusive OR of two GMP numbers.

gmp_perfect_square

(PHP 4 >= 4.0.4, PHP 5)

gmp_perfect_square -- Perfect square check

Description

bool **gmp_perfect_square** (resource a)

Returns **TRUE** if *a* is a prefect square, **FALSE** otherwise.

See also: **gmp_sqrt()**, **gmp_sqrtrm()**.

gmp_popcount

(PHP 4 >= 4.0.4, PHP 5)

gmp_popcount -- Population count

Description

int **gmp_popcount** (resource a)

Return the population count of *a*.

gmp_pow

(PHP 4 >= 4.0.4, PHP 5)

gmp_pow -- Raise number into power

Description

resource **gmp_pow** (resource base, int exp)

Raise *base* into power *exp*. The case of 0^0 yields 1. *exp* cannot be negative.

gmp_powm

(PHP 4 >= 4.0.4, PHP 5)

gmp_powm -- Raise number into power with modulo

Description

resource **gmp_powm** (resource base, resource exp, resource mod)

Calculate (*base* raised into power *exp*) modulo *mod*. If *exp* is negative, result is undefined.

gmp_prob_prime

(PHP 4 >= 4.0.4, PHP 5)

gmp_prob_prime -- Check if number is "probably prime"

Description

int **gmp_prob_prime** (resource a [, int reps])

If this function returns 0, *a* is definitely not prime. If it returns 1, then *a* is "probably" prime. If it returns 2, then *a* is surely prime. Reasonable values of *reps* vary from 5 to 10 (default being 10); a higher value lowers the probability for a non-prime to pass as a "probable" prime.

The function uses Miller-Rabin's probabilistic test.

gmp_random

(PHP 4 >= 4.0.4, PHP 5)

gmp_random -- Random number

Description

resource **gmp_random** (int limiter)

Generate a random number. The number will be up to *limiter* words long. If *limiter* is negative, negative numbers are generated.

gmp_scan0

(PHP 4 >= 4.0.4, PHP 5)

gmp_scan0 -- Scan for 0

Description

int **gmp_scan0** (resource a, int start)

Scans *a*, starting with bit *start*, towards more significant bits, until the first clear bit is found. Returns the index of the found bit.

gmp_scan1

(PHP 4 >= 4.0.4, PHP 5)

gmp_scan1 -- Scan for 1

Description

int **gmp_scan1** (resource a, int start)

Scans *a*, starting with bit *start*, towards more significant bits, until the first set bit is found. Returns the index of the found bit.

gmp_setbit

(PHP 4 >= 4.0.4, PHP 5)

gmp_setbit -- Set bit

Description

resource **gmp_setbit** (resource &a, int index [, bool set_clear])

Sets bit *index* in *a*. *set_clear* defines if the bit is set to 0 or 1. By default the bit is set to 1.

gmp_sign

(PHP 4 >= 4.0.4, PHP 5)

gmp_sign -- Sign of number

Description

int **gmp_sign** (resource a)

Return sign of *a* - 1 if *a* is positive and -1 if it's negative.

gmp_sqrt

(PHP 4 >= 4.0.4, PHP 5)

gmp_sqrt -- Square root

Description

resource **gmp_sqrt** (resource a)

Calculates square root of *a*.

gmp_sqrtrem

(PHP 4 >= 4.0.4, PHP 5)

gmp_sqrtrem -- Square root with remainder

Description

array **gmp_sqrtrem** (resource a)

Returns array where first element is the integer square root of *a* (see also **gmp_sqrt()**), and the second is the remainder (i.e., the difference between *a* and the first element squared).

Ejemplo 1. gmp_sqrtrem() example

```
<?php
list($sqrt1, $sqrt1rem) = gmp_sqrtrem("9");
list($sqrt2, $sqrt2rem) = gmp_sqrtrem("7");
list($sqrt3, $sqrt3rem) = gmp_sqrtrem("1048576");

echo gmp_strval($sqrt1) . ", " . gmp_strval($sqrt1rem) . "\n";
echo gmp_strval($sqrt2) . ", " . gmp_strval($sqrt2rem) . "\n";
echo gmp_strval($sqrt3) . ", " . gmp_strval($sqrt3rem) . "\n";
?>
```

The printout of the above program will be:

```
3, 0
2, 3
1024, 0
```

gmp_strval

(PHP 4 >= 4.0.4, PHP 5)

gmp_strval -- Convert GMP number to string

Description

string **gmp_strval** (resource gmpnumber [, int base])

Convert GMP number to string representation in base *base*. The default base is 10. Allowed values for the base are from 2 to 36.

Ejemplo 1. Converting GMP number to string

```
<?php
    $a = gmp_init("0x41682179fbf5");
    printf ("Decimal: %s, 36-based: %s", gmp_strval($a), gmp_strval($a,36));
?>
```

gmp_sub

(PHP 4 >= 4.0.4, PHP 5)

gmp_sub -- Subtract numbers

Description

resource **gmp_sub** (resource a, resource b)

Subtract *b* from *a* and returns the result.

gmp_xor

(PHP 4 >= 4.0.4, PHP 5)

gmp_xor -- Logical XOR

Description

resource **gmp_xor** (resource a, resource b)

Calculates logical exclusive OR (XOR) of two GMP numbers.

XLIII. Funciones HTTP

Introducción

Estas funciones le permiten manipular la salida que se envía de vuelta al navegador remoto al nivel del protocolo HTTP.

Requirimientos

No se necesitan bibliotecas externas para construir esta extensión

---

Instalación

No se necesita ninguna instalación para usar estas funciones, son parte del núcleo de PHP.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Esta extensión no tiene ninguna constante definida.

Tabla de contenidos

header

(PHP 3, PHP 4 , PHP 5)

header -- Enviar una cabecera HTTP pura

Descripción

void **header** (string cadena [, bool reemplazar [, int cod_respuesta_http]])

La función **header()** es usada para enviar cabeceras HTTP puras. Consulte la especificación HTTP/1.1 para más información sobre las cabeceras HTTP.

El parámetro opcional *reemplazar* indica si la cabecera debe reemplazar una cabecera previa semejante, o si debe agregar una segunda cabecera del mismo tipo. Por defecto esta función procede a reemplazar, pero si pasa **`FALSE`** como el segundo argumento, puede obligar a que se envíen múltiples cabeceras del mismo tipo. Por ejemplo:

```
<?php
header('WWW-Authenticate: Negotiate');
header('WWW-Authenticate: NTLM', false);
?>
```

El segundo parámetro opcional, *cod_respuesta_http*, obliga a que el código de respuesta HTTP sea el valor especificado. (Este parámetro se encuentra disponible a partir de PHP 4.3.0.)

Existen dos llamadas de cabecera que son casos especiales. El primero es una cabecera que comience con la cadena "*HTTP/*" (no es importante la diferencia entre mayúsculas y minúsculas), la cual será usada para elegir el código de status HTTP a enviar. Por ejemplo, si ha configurado a Apache para que use un script PHP a la hora de gestionar peticiones por archivos inexistentes (usando la directiva *ErrorDocument*), puede que quiera asegurarse de que su script genere el código de status apropiado.

```
<?php
header("HTTP/1.0 404 Not Found");
?>
```

> **Nota:** La línea de cabecera con el status HTTP será siempre la primera en ser enviada al cliente, independientemente de que la llamada a **header()** correspondiente sea la primera o no. El status puede ser sobrescrito llamando **header()** con una nueva línea de status en cualquier momento, a menos que las cabeceras HTTP ya hayan sido enviadas.
>
> **Nota:** En PHP 3, esto funciona únicamente cuando PHP ha sido compilado como un

módulo de Apache. Puede conseguir el mismo efecto usando la cabecera *Status*.

```
<?php
header("Status: 404 Not Found");
?>
```

El segundo caso especial es la cabecera "Location:". No solo envía esta cabecera de vuelta al navegador, sino que también devuelve un código de status *REDIRECT* (302) al navegador a menos que algún código de status *3xx* haya sido enviado ya.

```
<?php
header("Location: http://www.example.com/"); /* Redirigir al navegador */

/* Asegurarse de que no se ejecute el codigo adicional cuando se redireccione. */
exit;
?>
```

Nota: HTTP/1.1 requiere una URI absoluta como argumento a Location: incluyendo el esquema, el nombre del host y una ruta absoluta, aunque algunos clientes aceptan URIs relativas. Usualmente puede usar *$_SERVER['HTTP_HOST']*, *$_SERVER['PHP_SELF']* y **dirname()** para construir una URI absoluta a partir de una relativa:

```
<?php
header("Location: http://" . $_SERVER['HTTP_HOST']
                     . dirname($_SERVER['PHP_SELF'])
                     . "/" . $url_relativa);
?>
```

Con frecuencia, los scripts PHP generan contenido dinámico que no debe ser almacenado en caché por el navegador del cliente o cualquier caché de proxy entre el servidor y el navegador del cliente. Muchos proxys y clientes pueden ser obligados a deshabilitar el uso de cachés con:

```
<?php
// Fecha en el pasado
header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");

// siempre modificado
header("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT");

// HTTP/1.1
header("Cache-Control: no-store, no-cache, must-revalidate");
header("Cache-Control: post-check=0, pre-check=0", false);

// HTTP/1.0
header("Pragma: no-cache");
?>
```

Nota: Puede que encuentre que sus páginas no son puestas en caché aun cuando no use todas las cabeceras enteriores. Existe un número de opciones que puede que se encuentren a disposición de los usuarios en sus navegadores para modificar sus comportamientos de caché predeterminados. Al enviar las anteriores cabeceras, usted debería sobrescribir cualquier parámetro de configuración que de otra forma podría estar causando que la salida de su script sea puesta en caché.

Adicionalmente, **session_cache_limiter()** y el parámetro de configuración *session.cache_limiter* pueden ser usados para generar automáticamente las cabeceras de uso de caché apropiadas cuando se están usando sesiones.

Recuerde que la función **header()** debe ser llamada antes de que cualquier salida sea enviada, ya sea mediante etiquetas HTML normales, líneas en blanco de un archivo, o desde PHP. Es un error bastante común interpretar código extra con **include()**, **require()**, o alguna otra función de acceso de archivos, y terminar con espacios o líneas en blanco que son impresas antes de un llamado a

header(). El mismo problema existe cuando se usa un archivo PHP/HTML único.

```
<html>
<?php
/* Esto produce un error. Note la salida anterior, que se realiza
 * antes de llamar a header() */
header('Location: http://www.example.com/');
?>
```

> **Nota:** A partir de PHP 4, puede usar búferes de salida para evitar este problema, con el efecto lateral de que toda su salida al navegador es colocada en un búfer en el servidor hasta que usted la envíe. Puede lograr esto usando **ob_start()** y **ob_end_flush()** en su script, o al definir la directiva de configuración *output_buffering* en su `php.ini` u otros archivos de configuración.

Si desea que el usuario reciba un cuadro de diálogo para almacenar los datos que usted envía, como en el caso de un archivo PDF generado al vuelo, puede usar la cabecera Content-Disposition para indicar un nombre de archivo recomendado y obligar a que el navegador despliegue un diálogo de guardado.

```
<?php
// Enviaremos un PDF
header('Content-type: application/pdf');

// Se va a llamar descarga.pdf
header('Content-Disposition: attachment; filename="descarga.pdf"');

// La fuente del PDF se encuentra en original.pdf
readfile('original.pdf');
?>
```

> **Nota:** Existe un problema en Microsoft Internet Explorer 4.01 que impide que esto funcione. No existe forma de evitar este inconveniente. Hay también un fallo en Microsoft Internet Explorer 5.5 que interfiere con esto, el cual puede ser resuelto al actualizarse a Service Pack 2 o superior.
>
> **Nota:** Si se encuentra habilitado el modo seguro, el uid del script es agregado a la sección *realm* de la cabecera *WWW-Authenticate* si define ésta cabecera (usada para Autenticación HTTP).

Vea también **headers_sent()**, **setcookie()**, y la sección sobre autenticación HTTP.

headers_list

(PHP 5)

headers_list -- Devuelve una lista de cabeceras de respuesta enviadas (o listas para ser enviadas)

Descripción

array **headers_list** (void)

headers_list() devolverá una matriz indexada numéricamente de cabeceras a ser enviadas al navegador / cliente. Para determinar si estas cabeceras ya han sido enviadas o no, use **headers_sent()**.

Ejemplo 1. Ejemplos de uso de headers_list()

```
<?php

/* setcookie() agrega una cabecera de respuesta propia */
setcookie('foo', 'bar');

/* Definir una cabecera de respuesta personalizada.
   Esta es ignorada por la mayoria de clientes */
header("X-Sample-Test: foo");

/* Especificar contenido de texto plano en nuestra respuesta */
header('Content-type: text/plain');

/* Que cabeceras van a ser enviadas? */
var_dump(headers_list());

?>
```

esto producirá la siguiente salida:

```
array(4) {
  [0]=>
  string(29) "X-Powered-By: PHP/5.0.0"
  [1]=>
  string(19) "Set-Cookie: foo=bar"
  [2]=>
  string(18) "X-Sample-Test: foo"
  [3]=>
  string(24) "Content-type: text/plain"
}
```

Vea también: **headers_sent()**, **header()**, y **setcookie()**.

headers_sent

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

headers_sent -- Chequea si se han enviado cabeceras, y dónde

Descripción

bool **headers_sent** ([string &archivo [, int &linea]])

headers_sent() devolverá **FALSE** si no se han enviado cabeceras HTTP, o **TRUE** de lo contrario. Si los parámetros opcionales *archivo* y *linea* son definidos, **headers_sent()** colocará el nombre de archivo y número de línea de las fuentes PHP en donde inició la salida en las variables *archivo* y *linea*.

No puede agregar más líneas de cabeceras usando la función **header()** una vez el bloque de cabeceras ha sido enviado. Usando esta función al menos puede prevenir la recepción de mensajes de error relacionados con cabeceras HTTP. Otra opción es usar Búferes de Salida.

> **Nota:** Los parámetros opcionales *archivo* y *linea* fueron añadidos en PHP 4.3.0.

Ejemplo 1. Ejemplos de uso de headers_sent()

```php
<?php

// Si no se han enviado cabeceras, enviar una
if (!headers_sent()) {
    header('Location: http://www.example.com/');
    exit;
}

// Un ejemplo del uso de las parametros opcionales archivo y linea, a
// partir de PHP 4.3.0.
// Note que $nombre_archivo y $num_linea son pasados para su uso posterior.
// No les asigne valores con anterioridad.
if (!headers_sent($nombre_archivo, $num_linea)) {
    header('Location: http://www.example.com/');
    exit;

// Probablemente quiera producir un error aqui.
} else {

    echo "Las cabeceras ya fueron enviadas en $nombre_archivo en la linea " .
         "$num_linea\nNo es posible redireccionar, por ahora por favor " .
         "pulse este <a href=\"http://www.example.com\">enlace</a> en su " .
         "lugar\n";
    exit;
}

?>
```

Vea también **ob_start()**, **trigger_error()**, y **header()** para una discusión más detallada de los temas involucrados.

setcookie

(PHP 3, PHP 4 , PHP 5)

setcookie -- Envía una cookie

Descripción

int **setcookie** (string name, string value, int expire, string path, string domain, int secure)

setcookie() define una cookie para ser enviada con el resto de la información de la cabecera. Las cookies deben enviarse *antes* de mandar cualquier otra cabecera (esta es una restricción de las cookies, no de PHP). Esto requiere que sitúe las llamadas a esta función antes de cualquier etiqueta <*html*> o <*head*>.

Todos los parámetros excepto *name* son opcionales. Si sólo se especifica el parámetro name, la cookie con ese nombre se borrará del cliente remoto. También puede sustituir cualquier parámetro por una cadena de texto vacía (*""*) y saltar así ese parámetro. Los parámetros *expire* y *secure* son números enteros y no se pueden saltar con una cadena de texto vacía. En su lugar utilice un cero (*0*). El parámetro *expire* es un entero de tiempo típico de UNIX tal como lo devuelven las funciones **time()** o **mktime()**. El parámetro *secure* indica que la cookie se debe transmitir única y exclusivamente sobre una conexión segura HTTPS.

Fallos habituales:

Las cookies no se hacen visibles hasta la siguiente carga de una página para la que la cookie deba estar visible.

Las llamadas múltiples a **setcookie()** en el mismo script se ejecutarán en orden inverso. Si está intentando borrar una cookie antes de insertar otra, debe situar la llamada de inserción antes de la de borrado.

A continuación se muestran algunos ejemplos::
Ejemplo 1. setcookie(), ejemplos

```
setcookie("TestCookie","Test Value");
setcookie("TestCookie",$value,time()+3600);  /* expire in 1 hour */
setcookie("TestCookie",$value,time()+3600,"/~rasmus/",".utoronto.ca",1);
```

Tenga en cuenta que el campo value de la cookie se codifica como URL (urlencode) automáticamente cuando envía la cookie. Cuando ésta se recibe, se descodifica autómaticamente y se asigna a una variable con el mismo nombre que el nombre de la cookie. Para ver el contenido de nuestra cookie de prueba en un script, simplemente utilice uno de los siguientes ejemplos:

```
echo $TestCookie;
echo $HTTP_COOKIE_VARS["TestCookie"];
```

También puede utilizar arrays de cookies empleando la notación de array en el nombre de la cookie. Esto tiene como efecto establecer tantas cookies como elementos de array, pero cuando el script recibe la cookie, se guardan los valores en un array con el nombre de la cookie:

```
setcookie( "cookie[three]", "cookiethree" );
setcookie( "cookie[two]", "cookietwo" );
setcookie( "cookie[one]", "cookieone" );
if ( isset( $cookie ) ) {
   while( list( $name, $value ) = each( $cookie ) ) {
          echo "$name == $value<br>\n";
   }
}
```

Para obtener más información sobre las cookies, consulte la especificación de cookies de Netscape, que se encuentra en http://wp.netscape.com/newsref/std/cookie_spec.html.

Microsoft Internet Explorer 4 con Service Pack 1 no funciona correctamente con las cookies que tienen asociado el parámetro path.

Netscape Communicator 4.05 y Microsoft Internet Explorer 3.x funcionan aparentemente de manera incorrecta cuando no se especifican los parámetros path y time.

setrawcookie

(PHP 5)

setrawcookie -- Send a cookie without urlencoding the cookie value

Description

bool **setrawcookie** (string name [, string value [, int expire [, string path [, string domain [, bool secure]]]]])

setrawcookie() is exactly the same as **setcookie()** except that the cookie value will not be automatically urlencoded when sent to the browser.

See also **header()**, **setcookie()** and the cookies section.

XLIV. Funciones para Hyperwave

Introducción

Hyperwave ha sido desarrollado en el IICM en Graz. Comenzó con el nombre Hyper-G y cambió a Hyperwave cuando fue comercializado (Si lo recuerdo bien, fue en 1996).

Hyperwave no es software gratuiro. La versión actual, 4.1, está disponible en www.hyperwave.com. Se puede solicitar gratuitamente una versión limitada (30 días).

Hyperwave es un sistema de información similar a una base de datos (HIS, Hyperwave Information Server - Servidor Hyperwave de Información). Su objetivo es el almacenamiento y manipulación de documentos. Un documento puede ser cualquier bloque posible de datos que también puede ser almacenado en un archivo. Cada documento se acompaña por su registro de objeto. El registro de objeto contiene metadatos para el documento. Los metadatos son una lista de atributos que pueden ser extendidos por el usuario. Ciertos atributos siempre son fijados por el servidor Hyperwave, otros pueden ser modificados por el usuario. Un atributo es un par nombre/valor de la forma nombre=valor. El registro completo del objeto tiene tantos de estos pares como guste el usuario. El nombre de un atributo no tiene porqué ser único, p. ej. un título puede aparecer varias veces en el registro de un objeto. Esto tiene sentido si se desea especificar un título en diferentes idiomas. En dicho caso existe la convención de que cada valor de título esté precedido por la abreviatura de dos letras del idioma, seguida por dos puntos, como p. ej. 'en:Title in English' o 'es:Título en Español'. Otros atributos tales como descripciones o palabras clave son canditatos potenciales a esta diferenciación. También se pueden reemplazar las abreviaturas de idioma por cualquier otra cadena siempre y cuando estén separadas por los dos puntos del resto del valor del atributo.

Cada registro de objeto tiene una representación nativa como cadena con cada par nombre/valor separado por una línea nueva. La extensión Hyperwave también conoce una segunda representación que consiste en un array asociativo donde el nombre del atributo es la clave. Los valores de atributo multilingües en sí mismos forman otro array asociativo donde la clave es la abreviatura del idioma. Realmente cualquier atributo múltiple forma una tabla asociativa donde la cadena a la izquierda de los dos puntos en el valor de atributo es la clave. (Esto no se ha implementado por completo. Sólo los atributos Title, Description y Keyword son tratados adecuadamente.)

Aparte de los documentos, todos los hiper-enlaces contenidos en un documento son almacenados también como registros de objeto. Cuando el documento sea insertado en la base de datos, los hiper-enlaces que haya en un documento serán borrados del mismo y almacenados como objetos individuales. El registro de objeto del enlace contiene información acerca de dónde comienza y dónde termina. Para recuperar el documento original se deberá recuperar el documento sin los enlaces y la lista de los mismos para reinsertarla (Las funciones **hw_pipedocument()** y **hw_gettext()** hacen esto para usted). La ventaja de separar los enlaces del documento es obvia. Una vez un documento al que apunta un enlace cambia de nombre, el enlace puede modificarse fácilmente. El documento que contiene el enlace no se ve afectado. Incluso se puede añadir un enlace a un documento sin alterarlo.

Decir que **hw_pipedocument()** y **hw_gettext()** hacen automáticamente la inserción de enlaces no es tan simple como suena. Insertar los enlaces implica una cierta jerarquía en los documentos. En un servidor web esto viene dado por el sistema de archivos, pero el Hyperwave tiene su propia jerarquía y los nombres no representan la posición de un objeto en dicha jerarquía. Por tanto, la creación de los enlaces precisa primeramente de realizar un mapeado entre el espacio de nombres y la jerarquía del Hyperwave y el espacio de nombres respectivo de una jerarquía de web. La diferencia fundamental entre Hyperwave y la web es la distinción clara entre nombres y jerarquía que se da en el primero. El nombre no contiene ninguna información sobre la posición del objeto en

la jerarquía. En la web, el nombre también contiene la información sobre la posición en la jerarquía del objeto. Esto nos lleva a dos posibles formas de mapeo. O bien se reflejan la jerarquía del Hyperwave y el nombre del objeto Hyperwave en el URL o sólo el nombre. Para facilitar las cosas, se utiliza el segundo método. El objeto Hyperwave de nombre 'mi_objeto' es mapeado a 'http://host/mi_objeto' sin importar dónde reside dentro de la jerarquía de Hyperwave. Un objeto con el nombre 'padre/mi_objeto' podría ser el hijo de 'mi_objeto' en la jerarquía Hyperwave, aunque en el espacio de nombres web aparezca justamente lo opuesto y el usuario pueda ser llevado a confusión. Esto sólo se puede evitar seleccionando nombres de objetos razonables.

Hecha esta decisión surge un segundo problema. ¿Cómo implicar al PHP? el URL http://host/mi_objeto no llamará a ningún script PHP a no ser que se le diga al servidor que lo transforme en p. ej. 'http://host/script_php3/mi_objeto' y que el 'script_php3' luego evalúe la variable $PATH_INFO y recupere el objeto con nombre 'mi_objeto' del servidor Hyperwave. Hay sólo un pequeño inconveniente que se puede resolver fácilmente. Cuando se reescribe cualquier URL no se permite el acceso a ningún otro documento en el servidor web. Un script de PHP para buscar en el servidor Hyperwave sería imposible. Por lo tanto se necesitará al menos una segunda regla de reescritura para que excluya ciertos URL, como los que empiecen p. ej. por http://host/Hyperwave. Básicamente esto sería compartir un espacio de nombres entre el servidor web y el servidor Hyperwave.

Los enlaces se insertan en los documentos basándose en el mecanismo citado más arriba.

Se vuelve más complicado si el PHP no se está ejecutando como módulo del servidor o como script CGI, sino que se ejecuta como aplicación, p. ej. para volcar el contenido del servidor de Hyperwave a un CD-ROM. En dicho caso tiene sentido mantener la jerarquía Hyperwave y mapearla en el sistema de archivos. Esto entra conflicto con los nombres de los objetos si estos reflejan su propia jerarquía (p. ej. eligiendo nombres que comienzan por '/'). Por tanto, la '/' tendrá que ser reemplazada por otro carácter, p. ej. '_' para continuar.

El protocolo de red para comunicarse con el servidor Hyperwave se denomina HG-CSP (Hyper-G Client/Server Protocol, Protocolo Hyper-G Cliente/Servidor). Está basado en mensajes que inician ciertas acciones, p. ej. obtener el registro de un objeto. En versiones anteriores del Servidor Hyperwave se distribuyeron dos clientes nativos (Harmony, Amadeus) para la comunicación con el servidor. Ambos desaparecieron cuando se comercializó el Hyperwave. Para sustituírlo se proporcionó el llamado wavemaster. El wavemaster es como un conversor de protocolo de HTTP a HG-CSP. La idea es realizar toda la administración de la base de datos y la visualización de documentos con una interfaz web. El wavemaster implementa una serie de posicionadores para acciones que permiten personalizar la interfaz. Dicho conjunto de posicionadores se denomina el Lenguaje PLACE. El PLACE no posee muchas de las características de un lenguaje de programación real y las extensiones al mismo únicamente alargan la lista de posicionadores. Esto ha obligado al uso de JavaScript que, en mi opinión, no hace la vida más fácil.

Añadir soporte de Hyperwave al PHP rellenaría el espacio que deja la falta de un lenguaje de programación que permita personalizar la interfaz. El PHP implementa todos los mensajes definidos en el HG-CSP pero además proporciona comandos más poderosos, p. ej. recuperar documentos completos.

El Hyperwave tiene su propia terminología para dar nombre a ciertos tipos de información. Esta ha sido ampliamente extendida y anulada. Casi todas las funciones operan en uno de los siguientes tipos de datos.

- ID de objeto: Un valor entero único paara cada objeto del servidor Hyperwave. También es uno de los atributos del registro de objeto (ObjectID). Los ID de objeto se usan generalmente como parámetros de entrada que especifican a un objeto.

- registro de objeto: Una cadena con pares atributo-valor con la forma atributo=valos. Los pares están separados unos de otros por un retorno de carro. Un registro de objeto puede convertirse fácilmente en una tabla (array) de objetos usando **hw_object2array()**. Varias funciones devuelven registros de objeto. Los nombres de dichas funciones terminan en obj.

- tabla de objetos: Una tabla asociativa con todos los atributos de un objeto. La clave es el nombre del atributo. Si un atributo aparece más de una vez en un registro de objeto será convertido en otra tabla asociativa o indizada. Los atributos que dependen del idioma (como el título, claves o descripción) se convertirán en una tabla asociativa con la abreviatura del idioma como clave. El resto de los atributos múltiples crearán una tabla indizada. Las funciones de PHP nunca devuelven tablas de objetos.

- hw_document: Este es un nuevo tipo de datos que almacena el documento actual, p. ej. HTML, PDF, etc. Está algo optimizado para documentos HTML pero puede usarse para cualquier formato.

Varias funciones que devuelven una tabla de registros de objeto también devuelven una tabla asociativa con información estadística sobre los mismos. La tabla es el último elemento del registro de objeto. La tabla estadística contiene los siguientes elementos:

Hidden

Número de registros de objeto con el atributo PresentationHints puesto a Hidden.

CollectionHead

Número de registros de objeto con el atributo PresentationHints puesto a CollectionHead.

FullCollectionHead

Número de registros de objeto con el atributo PresentationHints puesto a FullCollectionHead.

CollectionHeadNr

Índice a una tabla de regitros de objeto con el atributo PresentationHints puesto a CollectionHead.

FullCollectionHeadNr

Índice a una tabla de regitros de objeto con el atributo PresentationHints puesto a FullCollectionHead.

Total

Total: Número de registros de objeto.

Integración con Apache

La extensión Hyperwave se utiliza mejor cuando el PHP se compila como un módulo de Apache. En tal caso el servidor Hyperwave subyacente puede ser ocultado casi por completo de los usuarios si el Apache utiliza su motor de re-escritura. Las siguientes instrucciones explicarán esto.

Como el PHP con soporte Hyperwave incluído en el Apache se ha diseñado para sustituir la solución nativa de Hyperwave basada en Wavemaster, asumiré que el servidor Apache sólo sirve como interfaz web para el Hyperwave. Esto no es necesario, pero simplifica la configuración. El concepto es bastante sencillo. Primeramente necesita un script PHP que evalúe la variable *PATH_INFO* y que trate su valor como el nombre de un objeto Hyperwave. Llamemos a este script 'Hyperwave'. El URL http://nombre.servidor/Hyperwave/nombre_de_objeto devolvería entonces el objeto Hyperwave llamado 'nombre_de_objeto'. Dependiendo del tipo del objeto, así reaccionará el script. Si es una colección, probablemente devolverá un lista de hijos. Si es un documento devolverá el tipo MIME y el contenido. Se puede mejorar ligeramente si se usa el motor de re-escritura del Apache. Desde el punto de vista del usuario será más sencillo si el URL http://nombre.servidor/nombre de objeto devuelve el objeto. La regla de reescritura es muy sencilla:

```
RewriteRule ^/(.*) /usr/local/apache/htdocs/HyperWave/$1 [L]
```

Ahora todo URL apunta a un objeto en el servidor Hyperwave. Esto provoca un problema sencillo de resolver. No hay forma de ejecutar otro script, p. ej. para buscar, salvo el script 'Hyperwave'. Esto se puede solucionar con otra regla de reescritura como la siguiente:

```
RewriteRule ^/hw/(.*) /usr/local/apache/htdocs/hw/$1 [L]
```

Esta reservará el directorio `/usr/local/apache/htdocs/hw` para script adicionales y otros archivos. Sólo hay que asegurarse que esta regla se evalúa antes de la otra. Sólo hay un pequeño inconveniente: todos los objetos Hyperwave cuyo nombre comienza por 'hw/' serán ocultados. así que asegúrese que no utiliza dichos nombres. Si necesita más directorios, p. ej. para imágenes, simplemente añada más reglas o sitúe los archivos en un solo directorio. Por último, no olvide conectar el motor de re-escritura con

```
RewriteEngine on
```

Mi experiencia me ha demostrado que necesitará los siguientes scripts:

- para devolver el script en sí
- para permitir las búsquedas
- para identificarse
- para ajustar su perfil
- uno para cada función adicional como mostrar los atributos del objeto, mostrar información sobre usuarios, mostrar el estado del servidor, etc.

Pendientes

Aún hay varias cosas pendientes:

- hw_InsertDocument debe dividirse en **hw_InsertObject()** y **hw_PutDocument()**.
- Los nombres de algunas funciones aún no están fijados.
- Muchas funciones precisan la conexión actual como primer parámetro. Esto obliga a escribir mucho, lo cual no es con frecuencia necesario si sólo hay una conexión abierta. Una conexión por defecto mejoraría esto.

- La conversión de registro de objeto a tabla de objeto necesita manipular cualquier atributo múltiple.

Tabla de contenidos

hw_Mv -- mueve objetos
hw_New_Document -- crear nuevo documento
hw_Objrec2Array -- convierte atributos de registro de objeto a tabla de objetos
hw_Output_Document -- Prints hw_document
hw_pConnect -- hacer una conexión de base de datos permanente
hw_PipeDocument -- recupera cualquier documento
hw_Root -- ID del objeto raíz
hw_setlinkroot -- Set the id to which links are calculated
hw_stat -- Returns status string
hw_Unlock -- desbloquea objeto
hw_Who -- Lista de los usuarios actualmente conectados

hw_Array2Objrec

(PHP 3>= 3.0.4, PHP 4)

hw_Array2Objrec -- convierte atributos de tabla de objeto a registro de objeto

Descripción

strin **hw_array2objrec** (array tabla_objetos)

Convierte una *tabla_objetos* en un registro de objeto. Los atributos múltiples como 'Título' en distintos idiomas son tratados correctamente.

Vea también **hw_objrec2array()**.

hw_changeobject

(PHP 3>= 3.0.3, PHP 4)

hw_changeobject -- Changes attributes of an object (obsolete)

Description

void **hw_changeobject** (int link, int objid, array attributes)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

hw_Children

(PHP 3>= 3.0.3, PHP 4)

hw_Children -- id de objeto de los hijos

Descripción

array **hw_children** (int conexión, int IDobjeto)

Devuelve una tabla de id de objeto. Cada uno de ellos pertenece a un hijo de la colección cuyo ID es *IDobjeto*. La tabla contiene tanto los documentos como las colecciones hijas.

hw_ChildrenObj

(PHP 3>= 3.0.3, PHP 4)

hw_ChildrenObj -- registros de objeto de los hijos

Descripción

array **hw_childrenobj** (int conexión, int IDobjeto)

Devuelve una tabla de registros de objeto. Cada uno de ellos pertenece a un hijo de la colección cuyo ID es *IDobjeto*. La tabla contiene tanto los documentos como las colecciones hijas.

hw_Close

(PHP 3>= 3.0.3, PHP 4)

hw_Close -- cierra la conexión Hyperwave

Descripción

int **hw_close** (int conexión)

Devuelve **`FALSE`** si la conexión no es un índice válido de conexión, y **`TRUE`** en caso contrario. Cierra la conexión dada con el servidor de Hyperwave.

hw_Connect

(PHP 3>= 3.0.3, PHP 4)

hw_Connect -- abre una conexión

Descripción

int **hw_connect** (string servidor, int puerto, string usuario, string clave)

Abre una conexión con un servidor Hyperwave y devuelve un índice de conexión si hay éxito, o **`FALSE`** si la conexión no se pudo realizar. Cada argumento debe ser una cadena entrecomillada salvo el número de puerto. Los argumentos de *usuario* y *clave* son opcionales y pueden omitirse. En tal caso no se realizará identificación alguna con el servidor. Es similar a identificarse como el usuario 'anonymous'. Esta función devuelve un índice de conexión que se necesita para otras funciones Hyperwave. Puede tener varias conexiones abiertas a la vez. Recuerde que la clave no

está encriptada.

Vea también **hw_pConnect()**.

hw_connection_info

(PHP 3>= 3.0.3, PHP 4)

hw_connection_info -- Prints information about the connection to Hyperwave server

Description

void **hw_connection_info** (int link)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

hw_Cp

(PHP 3>= 3.0.3, PHP 4)

hw_Cp -- copia objetos

Descripción

int **hw_cp** (int conexión, array tabla_id_objeto, int id destino)

Copia los objetos cuyos id se especifican en el segundo parámetro a la colección identificada como *id destino*.

El valor devuelto es el número de objetos copiados.

Vea también **hw_mv()**.

hw_Deleteobject

(PHP 3>= 3.0.3, PHP 4)

hw_Deleteobject -- borra objetos

Descripción

int **hw_deleteobject** (int conexión, int objeto_a_borrar)

Borra el objeto con el id dado como segundo parámetro. Borrará todas las instancias del mismo.

Devuelve **TRUE** si no hay error o **FALSE** en caso contrario.

Vea también **hw_mv()**.

hw_DocByAnchor

(PHP 3>= 3.0.3, PHP 4)

hw_DocByAnchor -- id del objeto al que pertenece un enlace

Descripción

int **hw_docbyanchor** (int conexión, int IDenlace)

Devuelve el id de objeto del documento al que pertenece el *IDenlace*.

hw_DocByAnchorObj

(PHP 3>= 3.0.3, PHP 4)

hw_DocByAnchorObj -- registro de objeto al que pertenece un enlace

Descripción

string **hw_docbyanchorobj** (int conexión, int IDenlace)

Devuelve el registro de objeto del documento al que pertenece el *IDenlace*.

hw_Document_Attributes

(PHP 3>= 3.0.3, PHP 4)

hw_Document_Attributes -- Object record of hw_document

Description

string **hw_document_attributes** (int hw_document)

Returns the object record of the document.

For backward compatibility, **hw_documentattributes()** is also accepted. This is deprecated, however.

See also **hw_document_bodytag()**, and **hw_document_size()**.

hw_Document_BodyTag

(PHP 3>= 3.0.3, PHP 4)

hw_Document_BodyTag -- Body tag of hw_document

Description

string **hw_document_bodytag** (int hw_document [, string prefix])

Returns the BODY tag of the document. If the document is an HTML document the BODY tag should be printed before the document.

See also **hw_document_attributes()**, and **hw_document_size()**.

For backward compatibility, **hw_documentbodytag()** is also accepted. This is deprecated, however.

hw_Document_Content

(PHP 3>= 3.0.3, PHP 4)

hw_Document_Content -- Returns content of hw_document

Description

string **hw_document_content** (int hw_document)

Returns the content of the document. If the document is an HTML document the content is everything after the BODY tag. Information from the HEAD and BODY tag is in the stored in the object record.

See also **hw_document_attributes()**, **hw_document_size()**, and **hw_document_setcontent()**.

hw_Document_SetContent

(PHP 4)

hw_Document_SetContent -- Sets/replaces content of hw_document

Description

string **hw_document_setcontent** (int hw_document, string content)

Sets or replaces the content of the document. If the document is an HTML document the content is everything after the BODY tag. Information from the HEAD and BODY tag is in the stored in the object record. If you provide this information in the content of the document too, the Hyperwave server will change the object record accordingly when the document is inserted. Probably not a very good idea. If this functions fails the document will retain its old content.

See also **hw_document_attributes()**, **hw_document_size()**, and **hw_document_content()**.

hw_Document_Size

(PHP 3>= 3.0.3, PHP 4)

hw_Document_Size -- Size of hw_document

Description

int **hw_document_size** (int hw_document)

Returns the size in bytes of the document.

See also **hw_document_bodytag()**, and **hw_document_attributes()**.

For backward compatibility, **hw_documentsize()** is also accepted. This is deprecated, however.

hw_dummy

(PHP 3>= 3.0.3, PHP 4)

hw_dummy -- Hyperwave dummy function

Description

string **hw_dummy** (int link, int id, int msgid)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

hw_EditText

(PHP 3>= 3.0.3, PHP 4)

hw_EditText -- recupera documento de texto

Descripción

int **hw_edittext** (int conexión, int documento_hw)

Envía el documento de texto al servidor. El registro de objeto del documento no puede ser modificado mientras el documento es editado. Esta función sólo funcionará para objetos puros de texto. No abrirá ninguna conexión especial de datos y por tanto bloquea la conexión de control durante la transferencia.

Vea también **hw_PipeDocument()**, **hw_FreeDocument()**, **hw_DocumentBodyTag()**, **hw_DocumentSize()**, **hw_OutputDocument()**, **hw_GetText()**.

hw_Error

(PHP 3>= 3.0.3, PHP 4)

hw_Error -- número de error

Descripción

int **hw_error** (int conexión)

Devuelve el último número de error. Si el valor devuelto es 0 no ha habido errores. El error está relacionado con el último comando.

hw_ErrorMsg

(PHP 3>= 3.0.3, PHP 4)

hw_ErrorMsg -- devuelve un mensaje de error

Descripción

string **hw_errormsg** (int conexión)

Devuelve una cadena que contiene el último mensaje de error o 'No Error'. Si devuelve **FALSE** es que la función fracasó. El mensaje está relacionado con el último comando.

hw_Free_Document

(PHP 3>= 3.0.3, PHP 4)

hw_Free_Document -- libera un documento_hw

Descripción

int **hw_free_document** (int documento_hw)

Libera la memoria ocupada por el documento Hyperwave.

hw_GetAnchors

(PHP 3>= 3.0.3, PHP 4)

hw_GetAnchors -- id de objeto de los enlaces de un documento

Descripción

array **hw_getanchors** (int conexión, int IDobjeto)

Devuelve una tabla de id de objeto con los enlaces del documento cuyo ID es *IDobjeto*.

hw_GetAnchorsObj

(PHP 3>= 3.0.3, PHP 4)

hw_GetAnchorsObj -- registros de objeto de los enlaces de un documento

Descripción

array **hw_getanchorsobj** (int conexión, int IDobjeto)

Devuelve una tabla de registros de objeto con los enlaces del documento cuyo ID es *IDobjeto*.

hw_GetAndLock

(PHP 3>= 3.0.3, PHP 4)

hw_GetAndLock -- devuelve registro de objeto y lo bloquea

Descripción

string **hw_getandlock** (int conexión, int IDobjeto)

Devuelve el registro de objeto para el objeto con ID *IDobjeto*. También bloqueará el objeto, de modo que otros usuarios no podrán acceder al mismo hasta que sea desbloqueado.

Vea también **hw_Unlock()**, **hw_GetObject()**.

hw_GetChildColl

(PHP 3>= 3.0.3, PHP 4)

hw_GetChildColl -- id de objeto de colecciones hijas

Descripción

array **hw_getchildcoll** (int conexión, int IDobjeto)

Devuelve una tabla de id de objeto. Cada ID de objeto pertenece a una colección hija de la colección con ID *IDobjeto*. La función no devolverá documentos hijos.

Vea también **hw_GetChildren()**, **hw_GetChildDocColl()**.

hw_GetChildCollObj

(PHP 3>= 3.0.3, PHP 4)

hw_GetChildCollObj -- registros de objeto de colecciones hijas

Descripción

array **hw_getchildcollobj** (int conexión, int IDobjeto)

Devuelve una tabla de registros de objeto. Cada uno de ellos pertenece a una colección hija de la

colección con ID *IDobjeto*. La función no devolverá documentos hijos.

Vea también **hw_ChildrenObj()**, **hw_GetChildDocCollObj()**.

hw_GetChildDocColl

(PHP 3>= 3.0.3, PHP 4)

hw_GetChildDocColl -- id de objeto de documentos hijos de una colección

Descripción

array **hw_getchilddoccoll** (int conexión, int IDobjeto)

Devuelve una tabla de id de objeto para los documentos hijos de una colección.

Vea también **hw_GetChildren()**, **hw_GetChildColl()**.

hw_GetChildDocCollObj

(PHP 3>= 3.0.3, PHP 4)

hw_GetChildDocCollObj -- registros de objeto de documentos hijos de una colección

Descripción

array **hw_getchilddoccollobj** (int conexión, int IDobjeto)

Devuelve una tabla de registros de objeto para los documentos hijos de una colección.

Vea también **hw_ChildrenObj()**, **hw_GetChildCollObj()**.

hw_GetObject

(PHP 3>= 3.0.3, PHP 4)

hw_GetObject -- registro de objeto

Descripción

array **hw_getobject** (int conexión, [int|array] IDobjeto, string consulta)

Devuelve el registro de objeto para el objeto cuyo ID es *IDobjeto* si el segundo parámetro es un entero. Si es una tabla la función devolverá una tabla de registros de objeto. En tal caso, el último parámetro, que es una cadena de consulta, también es evaluado.

La cadena de consulta tiene la sintáxis siguiente:

<expr> ::= "(" <expr> ")" |

"!" <expr> | /* NO */

<expr> "||" <expr> | /* O */

<expr> "&&" <expr> | /* Y */

<atributo> <operador> <valor>

<atributo> ::= /* cualquier atributo (Título, Autor, TipoDocumento ...) */

<operador> ::= "=" | /* igual */

"<" | /* menor que (comparación de cadenas) */

">" | /* mayor que (comparación de cadenas) */

"~" /* expresión regular */

La consulta permite seleccionar elementos de la lista de objetos dada. Al contrario de otras funciones de búsqueda, esta consulta no puede utilizar atributos indizados. El número de registros de objeto devueltos depende de la consulta y de si está permitido el acceso al objeto.

Vea también **hw_GetAndLock()**, **hw_GetObjectByQuery()**.

hw_GetObjectByQuery

(PHP 3>= 3.0.3, PHP 4)

hw_GetObjectByQuery -- buscar objeto

Descripción

array **hw_getobjectbyquery** (int conexión, string consulta, int max_resultados)

Busca objetos en todo el servidor y devuelve una tabla de id de objeto. El número máximo de resultados será *max_resultados*. Si *max_resultados* vale -1, el número máximo de resultados es ilimitado.

La consulta funcionará sólo con atributos indizados.

Vea también **hw_GetObjectByQueryObj()**.

hw_GetObjectByQueryColl

(PHP 3>= 3.0.3, PHP 4)

hw_GetObjectByQueryColl -- buscar objeto en colección

Descripción

array **hw_getobjectbyquerycoll** (int conexión, int IDobjeto, string consulta, int max_resultados)

Busca objetos en la colección cuyo ID es *IDobjeto* y devuelve una tabla de ID de objeto. El número máximo de resultados será *max_resultados*. Si *max_resultados* vale -1, el número máximo de resultados es ilimitado.

La consulta funcionará sólo con atributos indizados.

Vea también **hw_GetObjectByQueryCollObj()**.

hw_GetObjectByQueryCollObj

(PHP 3>= 3.0.3, PHP 4)

hw_GetObjectByQueryCollObj -- buscar objeto en colección

Descripción

array **hw_getobjectbyquerycollobj** (int conexión, int IDobjeto, string consulta, int max_resultados)

Busca objetos en la colección cuyo ID es *IDobjeto* y devuelve una tabla de registros de objeto. El número máximo de resultados será *max_resultados*. Si *max_resultados* vale -1, el número máximo de resultados es ilimitado.

La consulta funcionará sólo con atributos indizados.

Vea también **hw_GetObjectByQueryColl()**.

hw_GetObjectByQueryObj

(PHP 3>= 3.0.3, PHP 4)

hw_GetObjectByQueryObj -- buscar objeto

Descripción

array **hw_getobjectbyqueryobj** (int conexión, string consulta, int max_resultados)

Busca objetos en todo el servidor y devuelve una tabla de registros de objeto. El número máximo de resultados será *max_resultados*. Si *max_resultados* vale -1, el número máximo de resultados es ilimitado.

La consulta funcionará sólo con atributos indizados.

Vea también **hw_GetObjectByQuery()**.

hw_GetParents

(PHP 3>= 3.0.3, PHP 4)

hw_GetParents -- id de objeto de los padres

Descripción

array **hw_getparents** (int conexión, int IDobjeto)

Devuelve una tabla indizada de id de objeto. Cada una pertenece a un padre del objeto con ID *IDobjeto*.

hw_GetParentsObj

(PHP 3>= 3.0.3, PHP 4)

hw_GetParentsObj -- registros de objeto de los padres

Descripción

array **hw_getparentsobj** (int conexión, int IDobjeto)

Devuelve una tabla indizada de registros de objeto junto a una tabla asociativa con información estadística sobre estos. La tabla asociativa es el último elemento de la tabla devuelta. Cada registro de objeto pertenece a un padre del objeto con ID *IDobjeto*.

hw_getrellink

(PHP 3>= 3.0.3, PHP 4)

hw_getrellink -- Get link from source to dest relative to rootid

Description

string **hw_getrellink** (int link, int rootid, int sourceid, int destid)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

hw_GetRemote

(PHP 3>= 3.0.3, PHP 4)

hw_GetRemote -- Obtiene un documento remoto

Descripción

int **hw_getremote** (int conexión, int IDobjeto)

Devuelve un documento remoto. Los documentos remotos en la notación de Hyperwave son los obtenidos de una fuente externa. Los documentos remotos típicos son páginas web externas o consultas a bases de datos. Para poder acceder a las fuentes externas a través de documentos

remotos, el Hyperwave presenta el HGI (Hyperwave Gateway Interface - Interfaz de Pasarela de Hyperwave) que es similar al CGI. Actualmente, sólo se puede acceder a servidores ftp y http y a algunas bases de datos. Llamar a **hw_GetRemote()** devuelve el documento de la fuente externa. Si desea usar esta función debe familiarizarse con los HGI. Debería considerar el usar PHP en lugar del Hyperwave para acceder a fuentes externas. Añadir soporte de bases de datos a través de una pasarela Hyperwave sería más difícil que hacerlo en PHP.

Vea también **hw_GetRemoteChildren()**.

hw_GetRemoteChildren

(PHP 3>= 3.0.3, PHP 4)

hw_GetRemoteChildren -- Obtiene el hijo del documento remoto

Descripción

int **hw_getremotechildren** (int conexión, string registro de objeto)

Devuelve el hijo de un documento remoto. Los hijos de documentos remotos son en sí mismos documentos remotos. Esto tiene sentido cuando se afina una consulta de bases de datos y se explica en la Guía de Programación de Hyperwave. Si el número de hijos es 1 la función devolverá el documento en sí mismo formateado con la Interfaz de Pasarela de Hyperwave (HGI). Si el número de hijos es mayor de 1 devolverá una tabla de registros de objeto, con cada uno posible candidato para otra llamada a **hw_GetRemoteChildren()**. Dichos registros de objeto son virtuales y no existen en el servidor Hyperwave, y por lo tanto no poseen un ID de objeto válido. La apariencia exacta de dicho registro de objeto depende del HGI. Si desea usar esta función deberá estar muy familiarizado con los HGI. También debería considerar el uso del PHP en lugar de Hyperwave para acceder a fuentes externas. Añadir soporte de bases de datos a través de una pasarela de Hyperwave resulta más difícil que hacerlo en PHP.

Vea también **hw_GetRemote()**.

hw_GetSrcByDestObj

(PHP 3>= 3.0.3, PHP 4)

hw_GetSrcByDestObj -- Devuelve los enlaces que apuntan al objeto

Descripción

array **hw_getsrcbydestobj** (int conexión, int IDobjeto)

Devuelve los registros de objeto de todos los enlaces que apuntan al objeto con ID *IDobjeto*. El objeto puede ser tanto un documento como un enlace de tipo destino.

Vea también **hw_GetAnchors()**.

hw_GetText

(PHP 3>= 3.0.3, PHP 4)

hw_GetText -- obtiene un documento de texto

Descripción

int **hw_gettext** (int conexión, int IDobjeto [, mixed IDraiz/prefijo])

Devuelve el documento con ID de objeto *IDobjeto*. Si el documento tiene enlaces que pueden ser insertados, serán insertados ahora. El parámetro opcional *IDraiz/prefijo* puede ser una cadena o un entero. Si es un entero determina la forma en que los enlaces se insertan en el documento. Por defecto es 0 y los enlaces se crean a partir del nombre del objeto de destino de los mismos. Esto es útil para aplicaciones web. Si un enlace apunta a un objeto con nombre 'pelicula_internet' el enlace HTML será <A HREF="/pelicula_internet">. La posición actual del objeto de destino en la jerarquía de documentos es despreciada. Tendrá que ajustar su navegador web para que reescriba dicho URL a, por ejemplo, '/mi_script.php3/pelicula_internet'. 'mi_script.php3' deberá evaluar $PATH_INFO y recuperar el documento. Todos los enlaces tendrán el prefijo '/mi_script.php3'. Si no desea este efecto puede fijar el parámetro opcional *IDraiz/prefijo* al prefijo que desee en su lugar. En este caso deberá ser una cadena.

Si el *IDraiz/prefijo* es un entero distinto de 0, el enlace se construye con todos los nombres de objeto comenzando con el objeto de id *IDraiz/prefijo*, separado por una barra relativa al objeto actual. Si por ejemplo el documento anterior 'pelicula_internet' está situado en 'a-b-c-pelicula_internet' donde '-' es el separador entre niveles jerárquicos en el servidor Hyperwave y el documento fuente está situado en 'a-b-d-fuente', el enlace HTML resultante sería: <A HREF="../c/pelicula_internet">. Esto es útil cuando desea bajarse el contenido completo del servidor al disco y mapear la jerarquía de documentos en el sistema de archivos.

Esta función sólo trabajará en documentos de texto puros. No se abrirá una conexión de datos especial y por tanto bloqueará la conexión de control durante la transferencia.

Vea también **hw_PipeDocument()**, **hw_FreeDocument()**, **hw_DocumentBodyTag()**, **hw_DocumentSize()**, **hw_OutputDocument()**.

hw_getusername

(PHP 3>= 3.0.3, PHP 4)

hw_getusername -- nombre del usuario actualmente conectado

Descripción

string **hw_getusername** (int conexión)

Devuelve el nombre de usuario de la conexión.

hw_Identify

(PHP 3>= 3.0.3, PHP 4)

hw_Identify -- identificarse como usuario

Descripción

int **hw_identify** (string nombre, string clave)

Le identifica como el usuario *nombre* y *clave*. La identificación sólo es válida para la sesión actual. No pienso que esta función sea necesaria con mucha frecuancia. En muchos casos será más fácil identificarse al abrir la conexión.

Vea también **hw_Connect()**.

hw_InCollections

(PHP 3>= 3.0.3, PHP 4)

hw_InCollections -- comprueba si los id de objeto están en las colecciones

Descripción

array **hw_incollections** (int conexión, array tabla_id_objeto, array tabla_id_colecciones, int colecciones_devueltas)

Comprueba si el conjunto de objetos (documentos o colecciones) especificado por el parámetro *tabla_id_objeto* es parte de las colecciones enumeradas en *tabla_id_colecciones*. Cuando el cuarto parámetro *colecciones_devueltas* es 0, el subconjunto de id de objeto que es parte de las colecciones (es decir, los documentos o colecciones hijos de una o más colecciones de id de colecciones o de sus subcolecciones, recursivamente) es devuelto en forma de tabla. Cuando el cuarto parámetro es 1, sin embargo, el conjunto de colecciones que tienen uno o más objetos de este subconjunto como hijos es devuelto como tabla. Esta opción permite a un cliente, p. ej., remarcar en una visualización gráfica la parte de la jerarquía de colecciones que contiene las coincidencias de una consulta previa.

hw_Info

(PHP 3>= 3.0.3, PHP 4)

hw_Info -- información sobre conexión

Descripción

string **hw_info** (int conexión)

Devuelve información sobre la conexión actual. La cadena devuelta tiene el siguiente formato: <Cadenaservidor>, <Anfitrión>, <Puerto>, <Usuario>, <Puerto del Usuario>, <Intercambio de bytes>

hw_InsColl

(PHP 3>= 3.0.3, PHP 4)

hw_InsColl -- insertar colección

Descripción

int **hw_inscoll** (int conexión, int IDobjeto, array tabla_objetos)

Inserta una nueva colección con los atributos de la *tabla_objetos* en la colección cuyo ID de objeto es *IDobjeto*.

hw_InsDoc

(PHP 3>= 3.0.3, PHP 4)

hw_InsDoc -- insertar documento

Descripción

int **hw_insdoc** (int conexión, int IDpadre, string registro_de_objeto, string texto)

Inserta un nuevo documento con los atributos del *registro_de_objeto* en la colección cuyo ID de objeto es *IDpadre*. Esta función inserta tanto un registro de objeto sólo como un registro de objeto y el texto puro en ASCII dado en *texto* si este es especificado. si desea insertar un documento general de cualquier tipo, utilice en su lugar **hw_insertdocument()**.

Vea también **hw_InsertDocument()**, **hw_InsColl()**.

hw_insertanchors

(PHP 4 >= 4.0.4)

hw_insertanchors -- Inserts only anchors into text

Description

string **hw_insertanchors** (int hwdoc, array anchorecs, array dest [, array urlprefixes])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

hw_InsertDocument

(PHP 3>= 3.0.3, PHP 4)

hw_InsertDocument -- subir cualquier objeto

Descripción

int **hw_insertdocument** (int conexión, int id_padre, int documento_hw)

Sube un documento a la colección dada por *id_padre*. El documento debe ser creado antes con la función **hw_NewDocument()**. Asegúrese que el registro de objeto del nuevo documento contenga al menos los atributos: Type, DocumentType, Title y Name (así, en inglés). Posiblemente desee fijar también el MimeType. La función devuelve la id de objeto del nuevo documento, o **`FALSE`**.

Vea también **hw_PipeDocument()**.

hw_InsertObject

(PHP 3>= 3.0.3, PHP 4)

hw_InsertObject -- inserta un registro de objeto

Descripción

int **hw_insertobject** (int conexión, string reg de objeto, string parametro)

Inserta un objeto en el servidor. Este puede consistir en cualquier objeto hyperwave válido. Vea la documentación sobre el HG-CSP si desea información detallada sobre cuáles tienen que ser los parámetros.

Nota: Si se desea insertar un enlace, el atributo Position siempre se fijará a un valor comienzo/final o a 'invisible'. Las posiciones invisibles se necesitan si la anotación no tiene enlace correspondiente en el texto anotado.

Vea también **hw_PipeDocument()**, **hw_InsertDocument()**, **hw_InsDoc()**, **hw_InsColl()**.

hw_mapid

(PHP 3>= 3.0.13, PHP 4)

hw_mapid -- Mapea in id global a un id virtual local

Descripción

int **hw_mapid** (int conexión, int id servidor, int id objeto)

Mapea un id de objeto global en un servidor hyperwave, incluso con aquellos a los que no se ha conectado con **hw_connect()**, sobre un id virtual de objeto. Este id virtual se puede utilizar como cualquier otro id de objeto, p. ej. para obtener el registro de objeto por medio de **hw_getobject()**. El id de servidor es la primera parte del id global de objeto (GOid) de aquel que es realmene el número IP expresado como entero.

Nota: Para usar esta función deberá activar el indicador F_DISTRIBUTED, que actualmenes sólo se puede fijar en tiempo de compilación desde hg_comm.c. Por defecto está inactivo. Lea el

comentario al principio de hg_comm.c

hw_Modifyobject

(PHP 3>= 3.0.7, PHP 4)

hw_Modifyobject -- modifica el registro de objeto

Descripción

int **hw_modifyobject** (int conexión, int objeto_a_cambiar, array eliminar, array añadir, int modo)

Este comando permite eliminar, añadir, o modificar atributos individuales de un registro de objeto. El objeto está especificado por el ID de objeto *objeto_a_cambiar*. La primera tabla, *eliminar*, es la lista de atributos a eliminar. La segunda tabla, *añadir*, es la lista de atributos a añadir. Para modificar un atributo, hay que borrar el antiguo y añadir uno nuevo. **hw_modifyobject()** siempre eliminará los atributos antes de añadir los nuevos excepto si el valor del atributo a eliminar no es una cadena o una tabla.

El último parámetro determina si la modificación se realiza de manera recursiva. 1 quiere decir que sea recursiva. Si alguno de los objetos no se puede modificar, será ignorado sin avisar. Incluso **hw_error()** podría no indicar un error aunque alguno de los objetos no pueda ser modificado.

Las claves de ambas tablas son los nombres de los atributos. El valor de cada elemento de la tabla puede ser una tabla, una cadena o cualquier otra cosa. Si es una tabla, cada valor de atributo se construye como la clave de cada elemento más dos puntos y el valor de cada elemento. Si es una cadena se toma como valor del atributo. Una cadena vacía producirá la supresión completa del atributo. Si el valor no es ni cadena ni tabla, sino otra cosa, p. ej. un entero, no se realizará operación alguna en el atributo. Esto es necesario se desea añadir un atributo completamente nuevo, no solamente un nuevo valor para un atributo existente. Si la tabla eliminar contuviera una cadena vacía para dicho atributo, este se intentaría suprimir, lo que fallaría porque este no existe. La siguiente adición de un nuevo valor para dicho atributo también fallará. Fijando el valor para dicho atributo p. ej. a 0 hará que ni siquiera se intente eliminar, pero funcionará la adición del mismo.

Si desea cambiar el atributo 'Nombre' con el valor actual 'libros' por el de 'artículos' deberá crear dos tablas y llamar a **hw_modifyobject()**.

Ejemplo 1. modificando un atributo

```
// $conexion es una conexión con el servidor Hyperwave
       // $idobj es la ID del objeto a modificar
       $tablasupr = array("Nombre" => "libros");
       $tablaanad = array("Nombre" => "artículos");
       $hw_modifyobject($conexion, $idobj, $tablasupr, $tablaanad);
```

Para borrar/añadir un par nombre=valor de/a el registro de objeto, simplemente pase la tabla eliminar/añadir y fije el último/tercer parámetro a tabla vacía. Si el atributo es el primero con dicho nombre que se añade, fije el valor del atributo en la tabla eliminar a un valor entero.

Ejemplo 2. añadiendo un atributo completamente nuevo

```
// $conexion es una conexión con el servidor Hyperwave
       // $idobj es la ID del objeto a modificar
       $tablasupr = array("Nombre" => 0);
       $tablaanad = array("Nombre" => "artículos");
       $hw_modifyobject($conexion, $idobj, $tablasupr, $tablaanad);
```

> **Nota:** Los atributos plurilingües, p. ej. 'Título', se pueden modificar de dos maneras. O bien proporcionando los valores de los atributos en su forma nativa 'lenguaje':'título', bien proporcionando una tabla con los elementos para cada lenguaje según se describe

más arriba. El ejemplo anterior podría quedar entonces:

Ejemplo 3. modificando el atributo Título

```
$tablasupr = array("Título" => "es:Libros");
       $tablaanad = array("Título" => "es:Artículos");
       $hw_modifyobject($conexion, $idobj, $tablasupr, $tablaanad);
```

o

Ejemplo 4. modificando el atributo Título

```
$tablasupr = array("Título" => array("es" => "Libros"));
       $tablaanad = array("Título" => array("es" => "Artículos", "ge"=>"Artikel"));
       $hw_modifyobject($conexion, $idobj, $tablasupr, $tablaanad);
```

Esto elimina el título español 'Libros' y añade el título español 'Artículos' y el título alemán 'Artikel'.

Ejemplo 5. eliminando atributos

```
$tablasupr = array("Título" => "");
       $tablaanad = array("Tñitulo" => "es:Artículos");
       $hw_modifyobject($conexion, $idobj, $tablasupr, $tablaanad);
```

Nota: Esto eliminará todos los atributos con el nombre 'Título' y añadirá un nuevo atributo 'Título'. Esto es útil cuando se desea eliminar atributos de forma recursiva.

Nota: Si necesita eliminar todos los atributos con un cierto nombre tendrá que pasar una cadena vacía como el valor del atributo.

Nota: Sólo los atributos 'Title', 'Description' y 'Keyword' (así, en inglés) manejarán de forma apropiada el prefijo de idioma. Si estos atributos no llevaran prefijo de idioma, se les asignariá el prefijo 'xx'.

Nota: El atributo 'Name' es bastante especial. En algunos casos no puede ser completamente eliminado. Obtendrá un mensaje de error 'Change of base attribute' ('Cambio de atributo base', no está muy claro cuando ocurre). Por tanto, tendrá siempre que añadir un nuevo atributo Name primero y luego eliminar el anterior.

Nota: No debe rodear esta función de llamadas a **hw_getandlock()** ni a **hw_unlock()**. **hw_modifyobject()** ya lo hace internamente.

Devuelve **TRUE** si no hay error o **FALSE** en caso contrario.

hw_Mv

(PHP 3>= 3.0.3, PHP 4)

hw_Mv -- mueve objetos

Descripción

int **hw_mv** (int conexión, array tabla de id de objeto, int id origen, int id destino)

Mueve los objetos cuyas id se especifican en el segundo parámetro desde la colección con id *id origen* hasta la colección con el id *id destino*. Si el id de destino es 0, los objetos serán disociados de la colección origen. Si esta fuese la última instancia del objeto, este sería borrado. Si desea borrar todas las instancias de una vez, utilice **hw_deleteobject()**.

El valor devuelto es el número de objetos movidos.

Vea también **hw_cp()**, **hw_deleteobject()**.

hw_New_Document

(PHP 3>= 3.0.3, PHP 4)

hw_New_Document -- crear nuevo documento

Descripción

int **hw_new_document** (string registro_de_objeto, string datos_documento, int tama_documento)

Devuelve un nuevo documento Hyperwave en el que los datos del documento están fijados a *datos_documento* y el registro de objeto a *registro_de_objeto*. La longitud de los *datos_documento* debe pasarse en *tama_documento*. Esta función no inserta el documento en el servidor Hyperwave.

Vea también **hw_FreeDocument()**, **hw_DocumentSize()**, **hw_DocumentBodyTag()**, **hw_OutputDocument()**, **hw_InsertDocument()**.

hw_Objrec2Array

(PHP 3>= 3.0.3, PHP 4)

hw_Objrec2Array -- convierte atributos de registro de objeto a tabla de objetos

Descripción

array **hw_objrec2array** (string registro_de_objeto)

Convierte un *registro_de_objeto* en una tabla de objetos. Las claves de la tabla resultante son los nombres de los atributos. Los atributos múltiples como 'Título' en distintos idiomas forman su propia tabla. Las claves de esta tabla son las partes a la izquierda de los dos puntos del valor del atributo. Actualmente, sólo los atributos 'Title', 'Description' y 'Keyword' (así, en inglés) son correctamente tratados. Otros atributos múltiples generan una tabla indizada. Actualmente, sólo el atributo 'Group' es tratado correctamente.

Vea también **hw_array2objrec()**.

hw_Output_Document

(PHP 3>= 3.0.3, PHP 4)

hw_Output_Document -- Prints hw_document

Description

int **hw_output_document** (int hw_document)

Prints the document without the BODY tag.

For backward compatibility, **hw_outputdocument()** is also accepted. This is deprecated, however.

hw_pConnect

(PHP 3>= 3.0.3, PHP 4)

hw_pConnect -- hacer una conexión de base de datos permanente

Descripción

int **hw_pconnect** (string servidor, int puerto, string usuario, string clave)

Devuelve un índice de conexión si hay éxito, o **FALSE** si la conexión no puede hacerse. Abre una conexión permanente a un servidor Hyperwave. Cada uno de los argumentos debe ser una cadena entrecomillada excepto el número de puerto. El argumento *usuario* y la *clave* son opcionales y pueden omitirse. En tal caso no se realizará ninguna identificación con el servidor. Es similar a la identificación anónima del usuario. Esta función devuelve un índice de conexión que se necesita para otras funciones de Hyperwave. Se pueden tener varias conexiones permanentes abiertas a la vez.

Vea también **hw_Connect()**.

hw_PipeDocument

(PHP 3>= 3.0.3, PHP 4)

hw_PipeDocument -- recupera cualquier documento

Descripción

int **hw_pipedocument** (int conexión, int IDobjeto)

Devuelve el documento Hyperwave cuyo ID de objeto es *IDobjeto*. Si el documento tiene enlaces que pueden ser insertados, deberán haberse insertado ya. El documento será transferido a través de una conexión de datos especial que no bloquea la conexión de control.

Vea también **hw_GetText()** para saber más sobre inserción de enlaces, **hw_FreeDocument()**, **hw_DocumentSize()**, **hw_DocumentBodyTag()**, **hw_OutputDocument()**.

hw_Root

(PHP 3>= 3.0.3, PHP 4)

hw_Root -- ID del objeto raíz

Descripción

int **hw_root** ( )

Devuelve la ID de objeto de la colección hiper-raíz. Actualmente siempre vale 0. La colección hija

de la hiper-raíz es la colección raíz del servidor al que se ha conectado.

hw_setlinkroot

(PHP 3>= 3.0.3, PHP 4)

hw_setlinkroot -- Set the id to which links are calculated

Description

void **hw_setlinkroot** (int link, int rootid)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

hw_stat

(PHP 3>= 3.0.3, PHP 4)

hw_stat -- Returns status string

Description

string **hw_stat** (int link)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

hw_Unlock

(PHP 3>= 3.0.3, PHP 4)

hw_Unlock -- desbloquea objeto

Descripción

int **hw_unlock** (int conexión, int IDobjeto)

Desbloquea un documento para que otros usuarios puedan acceder al mismo de nuevo.

Vea también **hw_GetAndLock()**.

hw_Who

(PHP 3>= 3.0.3, PHP 4)

hw_Who -- Lista de los usuarios actualmente conectados

Descripción

int **hw_who** (int conexión)

Devuelve una tabla de los usuarios actualmente conectados al servidor Hyperwave. Cada elemento de esta tabla contiene en sí mismo los elementos ID, nombre, sistema, onSinceDate (conectadoDesdeFecha), onSinceTime (conectadoDesdeHora), TotalTime (TiempoTotal) y self (propio). 'self' es 1 si esta línea corresponde al usuario que realizó la petición.

XLV. Hyperwave API Functions

Introducción

Hyperwave has been developed at IICM in Graz. It started with the name Hyper-G and changed to Hyperwave when it was commercialised (in 1996).

Hyperwave is not free software. The current version, 5.5, is available at http://www.hyperwave.com/. A time limited version can be ordered for free (30 days).

See also the Hyperwave module.

Hyperwave is an information system similar to a database (HIS, Hyperwave Information Server). Its focus is the storage and management of documents. A document can be any possible piece of data that may as well be stored in file. Each document is accompanied by its object record. The object record contains meta data for the document. The meta data is a list of attributes which can be extended by the user. Certain attributes are always set by the Hyperwave server, other may be modified by the user.

Requirimientos

Since 2001 there is a Hyperwave SDK available. It supports Java, JavaScript and C++. This PHP Extension is based on the C++ interface. In order to activate the hwapi support in PHP you will have to install the Hyperwave SDK first.

Instalación

After installing the Hyperwave SDK, configure PHP with *--with-hwapi[=DIR]*.

Integration with Apache

The integration with Apache and possible other servers is already described in the Hyperwave module which has been the first extension to connect a Hyperwave Server.

Configuración en tiempo de ejecución

El comportamiento de estas funciones está afectado por los valores definidos en `php.ini`.

Tabla 1. Hyperwave API configuration options

Name	Default	Changeable
hwapi.allow_persistent	"0"	PHP_INI_SYSTEM

For further details and definitions of the PHP_INI_* constants, see the Apéndice H.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Esta extensión no tiene ninguna constante definida.

Classes

The API provided by the HW_API extension is fully object oriented. It is very similar to the C++ interface of the Hyperwave SDK. It consist of the following classes.

- **HW_API**
- **HW_API_Object**
- **HW_API_Attribute**
- **HW_API_Error**
- **HW_API_Content**
- **HW_API_Reason**

Some basic classes like **HW_API_String**, **HW_API_String_Array**, etc., which exist in the Hyperwave SDK have not been implemented since PHP has powerful replacements for them.

Each class has certain method, whose names are identical to its counterparts in the Hyperwave SDK. Passing arguments to this function differs from all the other PHP extensions but is close to the C++ API of the HW SDK. Instead of passing several parameters they are all put into an associated array and passed as one parameter. The names of the keys are identical to those documented in the HW SDK. The common parameters are listed below. If other parameters are required they will be documented if needed.

- **objectIdentifier** The name or id of an object, e.g. "rootcollection", "0x873A8768 0x00000002".
- **parentIdentifier** The name or id of an object which is considered to be a parent.
- **object** An instance of class HW_API_Object.
- **parameters** An instance of class HW_API_Object.
- **version** The version of an object.
- **mode** An integer value determine the way an operation is executed.
- **attributeSelector** Any array of strings, each containing a name of an attribute. This is used if you retrieve the object record and want to include certain attributes.
- **objectQuery** A query to select certain object out of a list of objects. This is used to reduce the number of objects which was delivered by a function like **hw_api->children()** or **hw_api->find()**.

Nota: Methods returning **boolean** can return **TRUE**, **FALSE** or **HW_API_Error** object.

Tabla de contenidos

hw_api_attribute->key

(no version information, might be only in CVS)

hw_api_attribute->key -- Returns key of the attribute

Description

string **hw_api_attribute->key** (void)

Returns the name of the attribute.

See also **hwapi_attribute_value()**.

hw_api_attribute->langdepvalue

(no version information, might be only in CVS)

hw_api_attribute->langdepvalue -- Returns value for a given language

Description

string **hw_api_attribute->langdepvalue** (string language)

Returns the value in the given language of the attribute.

See also **hwapi_attribute_value()**.

hw_api_attribute->value

(no version information, might be only in CVS)

hw_api_attribute->value -- Returns value of the attribute

Description

string **hw_api_attribute->value** (void)

Returns the value of the attribute.

See also **hwapi_attribute_key()**, **hwapi_attribute_values()**.

hw_api_attribute->values

(no version information, might be only in CVS)

hw_api_attribute->values -- Returns all values of the attribute

Description

array **hw_api_attribute->values** (void)

Returns all values of the attribute as an array of strings.

See also **hwapi_attribute_value()**.

hw_api_attribute

(no version information, might be only in CVS)

hw_api_attribute -- Creates instance of class hw_api_attribute

Description

HW_API_Attribute **hw_api_attribute** ([string name [, string value]])

Creates a new instance of hw_api_attribute with the given name and value.

hw_api->checkin

(no version information, might be only in CVS)

hw_api->checkin -- Checks in an object

Description

bool **hw_api->checkin** (array parameter)

This function checks in an object or a whole hierarchy of objects. The parameters array contains the required element 'objectIdentifier' and the optional element 'version', 'comment', 'mode' and 'objectQuery'. 'version' sets the version of the object. It consists of the major and minor version separated by a period. If the version is not set, the minor version is incremented. 'mode' can be one of the following values:

HW_API_CHECKIN_NORMAL

> Checks in and commits the object. The object must be a document.

HW_API_CHECKIN_RECURSIVE

> If the object to check in is a collection, all children will be checked in recursively if they are documents. Trying to check in a collection would result in an error.

HW_API_CHECKIN_FORCE_VERSION_CONTROL

> Checks in an object even if it is not under version control.

HW_API_CHECKIN_REVERT_IF_NOT_CHANGED

> Check if the new version is different from the last version. Unless this is the case the object will be checked in.

HW_API_CHECKIN_KEEP_TIME_MODIFIED

> Keeps the time modified from the most recent object.

HW_API_CHECKIN_NO_AUTO_COMMIT

> The object is not automatically committed on check-in.

See also **hwapi_checkout()**.

hw_api->checkout

(no version information, might be only in CVS)

hw_api->checkout -- Checks out an object

Description

bool **hw_api->checkout** (array parameter)

This function checks out an object or a whole hierarchy of objects. The parameters array contains the required element 'objectIdentifier' and the optional element 'version', 'mode' and 'objectQuery'. 'mode' can be one of the following values:

HW_API_CHECKIN_NORMAL

Checks out an object. The object must be a document.

HW_API_CHECKIN_RECURSIVE

If the object to check out is a collection, all children will be checked out recursively if they are documents. Trying to check out a collection would result in an error.

See also **hwapi_checkin()**.

hw_api->children

(no version information, might be only in CVS)

hw_api->children -- Returns children of an object

Description

array **hw_api->children** (array parameter)

Retrieves the children of a collection or the attributes of a document. The children can be further filtered by specifying an object query. The parameter array contains the required elements 'objectIdentifier' and the optional elements 'attributeSelector' and 'objectQuery'.

The return value is an array of objects of type **HW_API_Object** or **HW_API_Error**.

See also **hwapi_parents()**.

hw_api_content->mimetype

(no version information, might be only in CVS)

hw_api_content->mimetype -- Returns mimetype

Description

string **hw_api_content->mimetype** (void)

Returns the mimetype of the content.

hw_api_content->read

(no version information, might be only in CVS)

hw_api_content->read -- Read content

Description

string **hw_api_content->read** (string buffer, integer len)

Reads *len* bytes from the content into the given buffer.

hw_api->content

(no version information, might be only in CVS)

hw_api->content -- Returns content of an object

Description

HW_API_Content **hw_api->content** (array parameter)

This function returns the content of a document as an object of type **hw_api_content**. The parameter array contains the required elements 'objectidentifier' and the optional element 'mode'. The mode can be one of the constants **`HW_API_CONTENT_ALLLINKS`**, **`HW_API_CONTENT_REACHABLELINKS`** or **`HW_API_CONTENT_PLAIN`**. **`HW_API_CONTENT_ALLLINKS`** means to insert all anchors even if the destination is not reachable. **`HW_API_CONTENT_REACHABLELINKS`** tells **hw_api_content()** to insert only reachable links and **`HW_API_CONTENT_PLAIN`** will lead to document without any links.

hw_api->copy

(no version information, might be only in CVS)

hw_api->copy -- Copies physically

Description

hw_api_object **hw_api->copy** (array parameter)

This function will make a physical copy including the content if it exists and returns the new object or an error object. The parameter array contains the required elements 'objectIdentifier' and 'destinationParentIdentifier'. The optional parameter is 'attributeSelector'`

See also **hwapi_move()**, **hwapi_link()**.

hw_api->dbstat

(no version information, might be only in CVS)

hw_api->dbstat -- Returns statistics about database server

Description

hw_api_object **hw_api->dbstat** (array parameter)

See also **hwapi_dcstat()**, **hwapi_hwstat()**, **hwapi_ftstat()**.

hw_api->dcstat

(no version information, might be only in CVS)

hw_api->dcstat -- Returns statistics about document cache server

Description

hw_api_object **hw_api->dcstat** (array parameter)

See also **hwapi_hwstat()**, **hwapi_dbstat()**, **hwapi_ftstat()**.

hw_api->dstanchors

(no version information, might be only in CVS)

hw_api->dstanchors -- Returns a list of all destination anchors

Description

array **hw_api->dstanchors** (array parameter)

Retrieves all destination anchors of an object. The parameter array contains the required element 'objectIdentifier' and the optional elements 'attributeSelector' and 'objectQuery'.

See also **hwapi_srcanchors()**.

hw_api->dstofsrcanchor

(no version information, might be only in CVS)

hw_api->dstofsrcanchor -- Returns destination of a source anchor

Description

hw_api_object **hw_api->dstofsrcanchor** (array parameter)

Retrieves the destination object pointed by the specified source anchors. The destination object can either be a destination anchor or a whole document. The parameters array contains the required element 'objectIdentifier' and the optional element 'attributeSelector'.

See also **hwapi_srcanchors()**, **hwapi_dstanchors()**, **hwapi_objectbyanchor()**.

hw_api_error->count

(no version information, might be only in CVS)

hw_api_error->count -- Returns number of reasons

Description

int **hw_api_error->count** (void)

Returns the number of error reasons.

See also **hwapi_error_reason()**.

hw_api_error->reason

(no version information, might be only in CVS)

hw_api_error->reason -- Returns reason of error

Description

HW_API_Reason **hw_api_error->reason** (void)

Returns the first error reason.

See also **hwapi_error_count()**.

hw_api->find

(no version information, might be only in CVS)

hw_api->find -- Search for objects

Description

array **hw_api->find** (array parameter)

This functions searches for objects either by executing a key or/and full text query. The found objects can further be filtered by an optional object query. They are sorted by their importance. The second search operation is relatively slow and its result can be limited to a certain number of hits. This allows to perform an incremental search, each returning just a subset of all found documents, starting at a given index. The parameter array contains the 'keyquery' or/and 'fulltextquery' depending on who you would like to search. Optional parameters are 'objectquery', 'scope', 'languages' and 'attributeselector'. In case of an incremental search the optional parameters 'startIndex', numberOfObjectsToGet' and 'exactMatchUnit' can be passed.

hw_api->ftstat

(no version information, might be only in CVS)

hw_api->ftstat -- Returns statistics about fulltext server

Description

hw_api_object **hw_api->ftstat** (array parameter)

See also **hwapi_dcstat()**, **hwapi_dbstat()**, **hwapi_hwstat()**.

hwapi_hgcsp

(no version information, might be only in CVS)

hwapi_hgcsp -- Returns object of class hw_api

Description

HW_API **hwapi_hgcsp** (string hostname [, int port])

Opens a connection to the Hyperwave server on host *hostname*. The protocol used is HGCSP. If you do not pass a port number, 418 is used.

See also **hwapi_hwtp()**.

hw_api->hwstat

(no version information, might be only in CVS)

hw_api->hwstat -- Returns statistics about Hyperwave server

Description

hw_api_object **hw_api->hwstat** (array parameter)

See also **hwapi_dcstat()**, **hwapi_dbstat()**, **hwapi_ftstat()**.

hw_api->identify

(no version information, might be only in CVS)

hw_api->identify -- Log into Hyperwave Server

Description

bool **hw_api->identify** (array parameter)

Logs into the Hyperwave Server. The parameter array must contain the elements 'username' and 'password'.

The return value will be an object of type **HW_API_Error** if identification failed or TRUE if it was successful.

hw_api->info

(no version information, might be only in CVS)

hw_api->info -- Returns information about server configuration

Description

array **hw_api->info** (array parameter)

See also **hwapi_dcstat()**, **hwapi_dbstat()**, **hwapi_ftstat()**, **hwapi_hwstat()**.

hw_api->insert

(no version information, might be only in CVS)

hw_api->insert -- Inserts a new object

Description

hw_api_object **hw_api->insert** (array parameter)

Insert a new object. The object type can be user, group, document or anchor. Depending on the type other object attributes has to be set. The parameter array contains the required elements 'object' and 'content' (if the object is a document) and the optional parameters 'parameters', 'mode' and 'attributeSelector'. The 'object' must contain all attributes of the object. 'parameters' is an object as well holding further attributes like the destination (attribute key is 'Parent'). 'content' is the content of the document. 'mode' can be a combination of the following flags:

HW_API_INSERT_NORMAL

The object in inserted into the server.

HW_API_INSERT_FORCE_VERSION_CONTROL

HW_API_INSERT_AUTOMATIC_CHECKOUT

HW_API_INSERT_PLAIN

HW_API_INSERT_KEEP_TIME_MODIFIED

HW_API_INSERT_DELAY_INDEXING

See also **hwapi_replace()**.

hw_api->insertanchor

(no version information, might be only in CVS)

hw_api->insertanchor -- Inserts a new object of type anchor

Description

hw_api_object **hw_api->insertanchor** (array parameter)

This function is a shortcut for **hwapi_insert()**. It inserts an object of type anchor and sets some of the attributes required for an anchor. The parameter array contains the required elements 'object' and 'documentIdentifier' and the optional elements 'destinationIdentifier', 'parameter', 'hint' and 'attributeSelector'. The 'documentIdentifier' specifies the document where the anchor shall be inserted. The target of the anchor is set in 'destinationIdentifier' if it already exists. If the target does not exists the element 'hint' has to be set to the name of object which is supposed to be inserted later. Once it is inserted the anchor target is resolved automatically.

See also **hwapi_insertdocument()**, **hwapi_insertcollection()**, **hwapi_insert()**.

hw_api->insertcollection

(no version information, might be only in CVS)

hw_api->insertcollection -- Inserts a new object of type collection

Description

hw_api_object **hw_api->insertcollection** (array parameter)

This function is a shortcut for **hwapi_insert()**. It inserts an object of type collection and sets some of the attributes required for a collection. The parameter array contains the required elements 'object' and 'parentIdentifier' and the optional elements 'parameter' and 'attributeSelector'. See **hwapi_insert()** for the meaning of each element.

See also **hwapi_insertdocument()**, **hwapi_insertanchor()**, **hwapi_insert()**.

hw_api->insertdocument

(no version information, might be only in CVS)

hw_api->insertdocument -- Inserts a new object of type document

Description

hw_api_object **hw_api->insertdocument** (array parameter)

This function is a shortcut for **hwapi_insert()**. It inserts an object with content and sets some of the attributes required for a document. The parameter array contains the required elements 'object', 'parentIdentifier' and 'content' and the optional elements 'mode', 'parameter' and 'attributeSelector'. See **hwapi_insert()** for the meaning of each element.

See also **hwapi_insert()** **hwapi_insertanchor()**, **hwapi_insertcollection()**.

hw_api->link

(no version information, might be only in CVS)

hw_api->link -- Creates a link to an object

Description

bool **hw_api->link** (array parameter)

Creates a link to an object. Accessing this link is like accessing the object to links points to. The parameter array contains the required elements 'objectIdentifier' and 'destinationParentIdentifier'. 'destinationParentIdentifier' is the target collection.

The function returns **TRUE** on success or an error object.

See also **hwapi_copy()**.

hw_api->lock

(no version information, might be only in CVS)

hw_api->lock -- Locks an object

Description

bool **hw_api->lock** (array parameter)

Locks an object for exclusive editing by the user calling this function. The object can be only unlocked by this user or the system user. The parameter array contains the required element 'objectIdentifier' and the optional parameters 'mode' and 'objectquery'. 'mode' determines how an object is locked. **HW_API_LOCK_NORMAL** means, an object is locked until it is unlocked. **HW_API_LOCK_RECURSIVE** is only valid for collection and locks all objects within the collection and possible subcollections. **HW_API_LOCK_SESSION** means, an object is locked only as long as the session is valid.

See also **hwapi_unlock()**.

hw_api->move

(no version information, might be only in CVS)

hw_api->move -- Moves object between collections

Description

bool **hw_api->move** (array parameter)

See also **hw_objrec2array()**.

hw_api_content

(no version information, might be only in CVS)

hw_api_content -- Create new instance of class hw_api_content

Description

HW_API_Content **hw_api_content** (string content, string mimetype)

Creates a new content object from the string *content*. The mimetype is set to *mimetype*.

hw_api_object->assign

(no version information, might be only in CVS)

hw_api_object->assign -- Clones object

Description

bool **hw_api_object->assign** (array parameter)

Clones the attributes of an object.

hw_api_object->attreditable

(no version information, might be only in CVS)

hw_api_object->attreditable -- Checks whether an attribute is editable

Description

bool **hw_api_object->attreditable** (array parameter)

hw_api_object->count

(no version information, might be only in CVS)

hw_api_object->count -- Returns number of attributes

Description

int **hw_api_object->count** (array parameter)

hw_api_object->insert

(no version information, might be only in CVS)

hw_api_object->insert -- Inserts new attribute

Description

bool **hw_api_object->insert** (HW_API_Attribute attribute)

Adds an attribute to the object. Returns **TRUE** on success and otherwise **FALSE**.

See also **hwapi_object_remove()**.

hw_api_object

(no version information, might be only in CVS)

hw_api_object -- Creates a new instance of class hw_api_object

Description

hw_api_object **hw_api_object** (array parameter)

See also **hwapi_lock()**.

hw_api_object->remove

(no version information, might be only in CVS)

hw_api_object->remove -- Removes attribute

Description

bool **hw_api_object->remove** (string name)

Removes the attribute with the given name. Returns **TRUE** on success and otherwise **FALSE**.

See also **hwapi_object_insert()**.

hw_api_object->title

(no version information, might be only in CVS)

hw_api_object->title -- Returns the title attribute

Description

string **hw_api_object->title** (array parameter)

hw_api_object->value

(no version information, might be only in CVS)

hw_api_object->value -- Returns value of attribute

Description

string **hw_api_object->value** (string name)

Returns the value of the attribute with the given name or **FALSE** if an error occurred.

hw_api->object

(no version information, might be only in CVS)

hw_api->object -- Retrieve attribute information

Description

hw_api_object **hw_api->object** (array parameter)

This function retrieves the attribute information of an object of any version. It will not return the document content. The parameter array contains the required elements 'objectIdentifier' and the optional elements 'attributeSelector' and 'version'.

The returned object is an instance of class **HW_API_Object** on success or **HW_API_Error** if an error occurred.

This simple example retrieves an object and checks for errors.
Ejemplo 1. Retrieve an object

```
<?php
function handle_error($error)
{
  $reason = $error->reason(0);
  echo "Type: <b>";
  switch ($reason->type()) {
    case 0:
      echo "Error";
      break;
    case 1:
      echo "Warning";
      break;
    case 2:
      echo "Message";
      break;
  }
  echo "</b><br />\n";
  echo "Description: " . $reason->description("en") . "<br />\n";
}

function list_attr($obj)
{
  echo "<table>\n";
  $count = $obj->count();
  for ($i=0; $i<$count; $i++) {
    $attr = $obj->attribute($i);
    printf("<tr><td align=\"right\" bgcolor=\"#c0c0c0\"><b>%s</b></td><td bgcolor=\"#F0
            $attr->key(), $attr->value());
  }
  echo "</table>\n";
}

$hwapi = hwapi_hgcsp($g_config[HOSTNAME]);
$parms = array("objectIdentifier"=>"rootcollection", "attributeSelector"=>array("Title"
$root = $hwapi->object($parms);
if (get_class($root) == "HW_API_Error") {
  handle_error($root);
  exit;
}
list_attr($root);
?>
```

See also **hwapi_content()**.

hw_api->objectbyanchor

(no version information, might be only in CVS)

hw_api->objectbyanchor -- Returns the object an anchor belongs to

Description

hw_api_object **hw_api->objectbyanchor** (array parameter)

This function retrieves an object the specified anchor belongs to. The parameter array contains the required element 'objectIdentifier' and the optional element 'attributeSelector'.

See also **hwapi_dstofsrcanchor()**, **hwapi_srcanchors()**, **hwapi_dstanchors()**.

hw_api->parents

(no version information, might be only in CVS)

hw_api->parents -- Returns parents of an object

Description

array **hw_api->parents** (array parameter)

Retrieves the parents of an object. The parents can be further filtered by specifying an object query. The parameter array contains the required elements 'objectidentifier' and the optional elements 'attributeselector' and 'objectquery'.

The return value is an array of objects of type **HW_API_Object** or **HW_API_Error**.

See also **hwapi_children()**.

hw_api_reason->description

(no version information, might be only in CVS)

hw_api_reason->description -- Returns description of reason

Description

string **hw_api_reason->description** (void)

Returns the description of a reason

hw_api_reason->type

(no version information, might be only in CVS)

hw_api_reason->type -- Returns type of reason

Description

HW_API_Reason **hw_api_reason->type** (void)

Returns the type of a reason.

hw_api->remove

(no version information, might be only in CVS)

hw_api->remove -- Delete an object

Description

bool **hw_api->remove** (array parameter)

Removes an object from the specified parent. Collections will be removed recursively. You can pass

an optional object query to remove only those objects which match the query. An object will be deleted physically if it is the last instance. The parameter array contains the required elements 'objectidentifier' and 'parentidentifier'. If you want to remove a user or group 'parentidentifier' can be skipped. The optional parameter 'mode' determines how the deletion is performed. In normal mode the object will not be removed physically until all instances are removed. In physical mode all instances of the object will be deleted immediately. In removelinks mode all references to and from the objects will be deleted as well. In nonrecursive the deletion is not performed recursive. Removing a collection which is not empty will cause an error.

See also **hwapi_move()**.

hw_api->replace

(no version information, might be only in CVS)

hw_api->replace -- Replaces an object

Description

hw_api_object **hw_api->replace** (array parameter)

Replaces the attributes and the content of an object The parameter array contains the required elements 'objectIdentifier' and 'object' and the optional parameters 'content', 'parameters', 'mode' and 'attributeSelector'. 'objectIdentifier' contains the object to be replaced. 'object' contains the new object. 'content' contains the new content. 'parameters' contain extra information for HTML documents. HTML_Language is the letter abbreviation of the language of the title. HTML_Base sets the base attribute of the HTML document. 'mode' can be a combination of the following flags:

HW_API_REPLACE_NORMAL

The object on the server is replace with the object passed.

HW_API_REPLACE_FORCE_VERSION_CONTROL

HW_API_REPLACE_AUTOMATIC_CHECKOUT

HW_API_REPLACE_AUTOMATIC_CHECKIN

HW_API_REPLACE_PLAIN

HW_API_REPLACE_REVERT_IF_NOT_CHANGED

HW_API_REPLACE_KEEP_TIME_MODIFIED

See also **hwapi_insert()**.

hw_api->setcommittedversion

(no version information, might be only in CVS)

hw_api->setcommittedversion -- Commits version other than last version

Description

hw_api_object **hw_api->setcommittedversion** (array parameter)

Commits a version of a document. The committed version is the one which is visible to users with read access. By default the last version is the committed version.

See also **hwapi_checkin()**, **hwapi_checkout()**, **hwapi_revert()**.

hw_api->srcanchors

(no version information, might be only in CVS)

hw_api->srcanchors -- Returns a list of all source anchors

Description

array **hw_api->srcanchors** (array parameter)

Retrieves all source anchors of an object. The parameter array contains the required element 'objectIdentifier' and the optional elements 'attributeSelector' and 'objectQuery'.

See also **hwapi_dstanchors()**.

hw_api->srcsofdst

(no version information, might be only in CVS)

hw_api->srcsofdst -- Returns source of a destination object

Description

array **hw_api->srcsofdst** (array parameter)

Retrieves all the source anchors pointing to the specified destination. The destination object can either be a destination anchor or a whole document. The parameters array contains the required element 'objectIdentifier' and the optional element 'attributeSelector' and 'objectQuery'. The function returns an array of objects or an error.

See also **hwapi_dstofsrcanchor()**.

hw_api->unlock

(no version information, might be only in CVS)

hw_api->unlock -- Unlocks a locked object

Description

bool **hw_api->unlock** (array parameter)

Unlocks a locked object. Only the user who has locked the object and the system user may unlock an object. The parameter array contains the required element 'objectIdentifier' and the optional parameters 'mode' and 'objectquery'. The meaning of 'mode' is the same as in function **hwapi_lock ()**.

Returns **TRUE** on success or an object of class HW_API_Error.

See also **hwapi_lock()**.

hw_api->user

(no version information, might be only in CVS)

hw_api->user -- Returns the own user object

Description

hw_api_object **hw_api->user** (array parameter)

See also **hwapi_userlist()**.

hw_api->userlist

(no version information, might be only in CVS)

hw_api->userlist -- Returns a list of all logged in users

Description

array **hw_api->userlist** (array parameter)

See also **hwapi_user()**.

XLVI. Funciones InterBase

Tabla de contenidos

ibase_add_user

(PHP 4 >= 4.2.0, PHP 5)

ibase_add_user -- Add a user to a security database (only for IB6 or later)

Description

bool **ibase_add_user** (resource service_handle, string user_name, string password [, string first_name [, string middle_name [, string last_name]]])

PHP 4 uses *server*, *dba_user_name* and *dba_user_password* instead of *service_handle* parameter.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

See also **ibase_modify_user()** and **ibase_delete_user()**.

ibase_affected_rows

(PHP 5)

ibase_affected_rows -- Return the number of rows that were affected by the previous query

Description

int **ibase_affected_rows** ([resource link_identifier])

This function returns the number of rows that were affected by the previous query that was executed from within the transaction context specified by *link_identifier*. If *link_identifier* is a connection resource, its default transaction is used.

See also **ibase_query()** and **ibase_execute()**.

ibase_backup

(PHP 5)

ibase_backup -- Initiates a backup task in the service manager and returns immediately

Description

mixed **ibase_backup** (resource service_handle, string source_db, string dest_file [, int options [, bool verbose]])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ibase_blob_add

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

ibase_blob_add -- Add data into a newly created blob

Description

bool **ibase_blob_add** (resource blob_handle, string data)

ibase_blob_add() adds data into a blob created with **ibase_blob_create()**. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

See also **ibase_blob_cancel()**, **ibase_blob_close()**, **ibase_blob_create()** and **ibase_blob_import()**.

ibase_blob_cancel

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

ibase_blob_cancel -- Cancel creating blob

Description

bool **ibase_blob_cancel** (resource blob_handle)

This function will discard a BLOB created by **ibase_create_blob()** if it has not yet been closed by **ibase_blob_close()**. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

See also **ibase_blob_close()**, **ibase_blob_create()** and **ibase_blob_import()**.

ibase_blob_close

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

ibase_blob_close -- Close blob

Description

mixed **ibase_blob_close** (resource blob_handle)

This function closes a BLOB that has either been opened for reading by **ibase_open_blob()** or has been opened for writing by **ibase_create_blob()**. If the BLOB was being read, this function returns **TRUE** on success, if the BLOB was being written to, this function returns a string containing the BLOB id that has been assigned to it by the database. On failure, this function returns **FALSE**.

See also **ibase_blob_cancel()** and **ibase_blob_open()**.

ibase_blob_create

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

ibase_blob_create -- Create a new blob for adding data

Description

resource **ibase_blob_create** ([resource link_identifier])

ibase_blob_create() creates a new BLOB for filling with data. It returns a BLOB handle for later use with **ibase_blob_add()** or **FALSE** on failure.

See also **ibase_blob_add()**, **ibase_blob_cancel()**, **ibase_blob_close()** and **ibase_blob_import()**.

ibase_blob_echo

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

ibase_blob_echo -- Output blob contents to browser

Description

bool **ibase_blob_echo** (resource link_identifier, string blob_id)

bool **ibase_blob_echo** (string blob_id)

This function opens a BLOB for reading and sends its contents directly to standard output (the browser, in most cases). Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

See also **ibase_blob_open()**, **ibase_blob_close()** and **ibase_blob_get()**.

ibase_blob_get

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

ibase_blob_get -- Get len bytes data from open blob

Description

string **ibase_blob_get** (resource blob_handle, int len)

This function returns at most *len* bytes from a BLOB that has been opened for reading by **ibase_blob_open()**. Returns **FALSE** on failure.

```
<?php
    $sql       = "SELECT blob_value FROM table";
    $result    = ibase_query($sql);
    $data      = ibase_fetch_object($result);
    $blob_data = ibase_blob_info($data->BLOB_VALUE);
    $blob_hndl = ibase_blob_open($data->BLOB_VALUE);
    echo         ibase_blob_get($blob_hndl, $blob_data[0]);
?>
```

Whilst this example doesn't do much more than a 'ibase_blob_echo($data->BLOB_VALUE)' would do, it does show you how to get information into a $variable to manipulate as you please.

> **Nota:** It is not possible to read from a BLOB that has been opened for writing by **ibase_blob_create()**.

See also **ibase_blob_open()**, **ibase_blob_close()** and **ibase_blob_echo()**.

ibase_blob_import

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

ibase_blob_import -- Create blob, copy file in it, and close it

Description

string **ibase_blob_import** (resource link_identifier, resource file_handle)

string **ibase_blob_import** (resource file_handle)

This function creates a BLOB, reads an entire file into it, closes it and returns the assigned BLOB id. The file handle is a handle returned by **fopen()**. Returns **FALSE** on failure.

Ejemplo 1. ibase_blob_import() example

```
<?php
$dbh = ibase_connect($host, $username, $password);
$filename = '/tmp/bar';

$fd = fopen($filename, 'r');
if ($fd) {

    $blob = ibase_blob_import($dbh, $fd);
    fclose($fd);

    if (!is_string($blob)) {
        // import failed
    } else {
        $query = "INSERT INTO foo (name, data) VALUES ('$filename', ?)";
        $prepared = ibase_prepare($dbh, $query);
        if (!ibase_execute($prepared, $blob)) {
            // record insertion failed
        }
    }
} else {
    // unable to open the data file
}
?>
```

See also **ibase_blob_add()**, **ibase_blob_cancel()**, **ibase_blob_close()** and **ibase_blob_create()**.

ibase_blob_info

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

ibase_blob_info -- Return blob length and other useful info

Description

array **ibase_blob_info** (resource link_identifier, string blob_id)

array **ibase_blob_info** (string blob_id)

Returns an array containing information about a BLOB. The information returned consists of the length of the BLOB, the number of segments it contains, the size of the largest segment, and whether it is a stream BLOB or a segmented BLOB.

ibase_blob_open

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

ibase_blob_open -- Open blob for retrieving data parts

Description

resource **ibase_blob_open** (resource link_identifier, string blob_id)

resource **ibase_blob_open** (string blob_id)

ibase_blob_open() opens an existing BLOB for reading. It returns a BLOB handle for later use with **ibase_blob_get()** or **FALSE** on failure.

See also **ibase_blob_close()**, **ibase_blob_echo()** and **ibase_blob_get()**.

ibase_close

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

ibase_close --

Descripcion

ibase_close ( )

ibase_commit_ret

(PHP 5)

ibase_commit_ret -- Commit a transaction without closing it

Description

bool **ibase_commit_ret** ([resource link_or_trans_identifier])

If called without an argument, this function commits the default transaction of the default link. If the argument is a connection identifier, the default transaction of the corresponding connection will be committed. If the argument is a transaction identifier, the corresponding transaction will be committed. The transaction context will be retained, so statements executed from within this transaction will not be invalidated. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

ibase_commit

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

ibase_commit -- Commit a transaction

Description

bool **ibase_commit** ([resource link_or_trans_identifier])

If called without an argument, this function commits the default transaction of the default link. If the argument is a connection identifier, the default transaction of the corresponding connection will be committed. If the argument is a transaction identifier, the corresponding transaction will be committed. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

ibase_connect

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

ibase_connect --

Descripcion

ibase_connect ( )

ibase_db_info

(PHP 5)

ibase_db_info -- Request statistics about a database

Description

string **ibase_db_info** (resource service_handle, string db, int action [, int argument])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ibase_delete_user

(PHP 4 >= 4.2.0, PHP 5)

ibase_delete_user -- Delete a user from a security database (only for IB6 or later)

Description

bool **ibase_delete_user** (resource service_handle, string user_name)

PHP 4 uses *server*, *dba_user_name* and *dba_user_password* instead of *service_handle* parameter.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

See also **ibase_add_user()** and **ibase_modify_user()**.

ibase_drop_db

(PHP 5)

ibase_drop_db -- Drops a database

Description

bool **ibase_drop_db** ([resource connection])

This functions drops a database that was opened by either **ibase_connect()** or **ibase_pconnect()**. The database is closed and deleted from the server. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

See also **ibase_connect()** and **ibase_pconnect()**.

ibase_errcode

(PHP 5)

ibase_errcode -- Return an error code

Description

int **ibase_errcode** (void)

Returns the error code that resulted from the most recent InterBase function call. Returns **FALSE** if no error occurred.

See also **ibase_errmsg()**.

ibase_errmsg

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

ibase_errmsg -- Return error messages

Description

string **ibase_errmsg** (void)

Returns the error message that resulted from the most recent InterBase function call. Returns **FALSE** if no error occurred.

See also **ibase_errcode()**.

ibase_execute

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

ibase_execute --

Descripcion

ibase_execute ( )

ibase_fetch_assoc

(PHP 4 >= 4.3.0, PHP 5)

ibase_fetch_assoc -- Fetch a result row from a query as an associative array

Description

array **ibase_fetch_assoc** (resource result [, int fetch_flag])

ibase_fetch_assoc() returns an associative array that corresponds to the fetched row. Subsequent calls will return the next row in the result set, or **FALSE** if there are no more rows.

ibase_fetch_assoc() fetches one row of data from the *result*. If two or more columns of the result have the same field names, the last column will take precedence. To access the other column(s) of the same name, you either need to access the result with numeric indices by using **ibase_fetch_row ()** or use alias names in your query.

fetch_flag is a combination of the constants IBASE_TEXT and IBASE_UNIXTIME ORed together. Passing IBASE_TEXT will cause this function to return BLOB contents instead of BLOB ids. Passing IBASE_UNIXTIME will cause this function to return date/time values as Unix timestamps instead of as formatted strings.

See also **ibase_fetch_row()** and **ibase_fetch_object()**.

ibase_fetch_object

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

ibase_fetch_object -- Get an object from a InterBase database

Description

object **ibase_fetch_object** (resource result_id [, int fetch_flag])

Fetches a row as a pseudo-object from a *result_id* obtained either by **ibase_query()** or **ibase_execute()**.

```
<?php
    $dbh = ibase_connect($host, $username, $password);
    $stmt = 'SELECT * FROM tblname';
    $sth = ibase_query($dbh, $stmt);
    while ($row = ibase_fetch_object($sth)) {
        echo $row->email . "\n";
    }
    ibase_close($dbh);
?>
```

Subsequent calls to **ibase_fetch_object()** return the next row in the result set, or **FALSE** if there are no more rows.

fetch_flag is a combination of the constants IBASE_TEXT and IBASE_UNIXTIME ORed together. Passing IBASE_TEXT will cause this function to return BLOB contents instead of BLOB ids. Passing IBASE_UNIXTIME will cause this function to return date/time values as Unix timestamps instead of as formatted strings.

See also **ibase_fetch_row()** and **ibase_fetch_assoc()**.

ibase_fetch_row

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

ibase_fetch_row --

Descripcion

ibase_fetch_row ( )

ibase_field_info

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

ibase_field_info -- Get information about a field

Description

array **ibase_field_info** (resource result, int field_number)

Returns an array with information about a field after a select query has been run. The array is in the form of name, alias, relation, length, type.

```
<?php
    $rs = ibase_query("SELECT * FROM tablename");
    $coln = ibase_num_fields($rs);
    for ($i = 0; $i < $coln; $i++) {
        $col_info = ibase_field_info($rs, $i);
        echo "name: ". $col_info['name']. "\n";
        echo "alias: ". $col_info['alias']. "\n";
        echo "relation: ". $col_info['relation']. "\n";
        echo "length: ". $col_info['length']. "\n";
        echo "type: ". $col_info['type']. "\n";
    }
?>
```

See also: **ibase_num_fields()**.

ibase_free_event_handler

(PHP 5)

ibase_free_event_handler -- Cancels a registered event handler

Description

bool **ibase_free_event_handler** (resource event)

This function causes the registered event handler specified by *event* to be cancelled. The callback function will no longer be called for the events it was registered to handle. Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

See also **ibase_set_event_handler()**.

ibase_free_query

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

ibase_free_query --

Descripcion

ibase_free_query ( )

ibase_free_result

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

ibase_free_result --

Descripcion

ibase_free_result ( )

ibase_gen_id

(PHP 5)

ibase_gen_id -- Increments the named generator and returns its new value

Description

int **ibase_gen_id** (string generator [, int increment [, resource link_identifier]])

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ibase_maintain_db

(PHP 5)

ibase_maintain_db -- Execute a maintenance command on the database server

Description

bool **ibase_maintain_db** (resource service_handle, string db, int action [, int argument])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ibase_modify_user

(PHP 4 >= 4.2.0, PHP 5)

ibase_modify_user -- Modify a user to a security database (only for IB6 or later)

Description

bool **ibase_modify_user** (resource service_handle, string user_name, string password [, string first_name [, string middle_name [, string last_name]]])

PHP 4 uses *server*, *dba_user_name* and *dba_user_password* instead of *service_handle* parameter.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

See also **ibase_add_user()** and **ibase_delete_user()**.

ibase_name_result

(PHP 5)

ibase_name_result -- Assigns a name to a result set

Description

bool **ibase_name_result** (resource result, string name)

This function assigns a name to a result set. This name can be used later in UPDATE|DELETE ... WHERE CURRENT OF *name* statements. Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

```
<?php
 $result = ibase_query("SELECT field1,field2 FROM table FOR UPDATE");
    ibase_name_result($result, "my_cursor");

    $updateqry = ibase_prepare("UPDATE table SET field2 = ? WHERE CURRENT OF my_cursor"

    for ($i = 0; ibase_fetch_row($result); ++$i) {
        ibase_execute($updateqry, $i);
    }
?>
```

See also **ibase_prepare()** and **ibase_execute()**.

ibase_num_fields

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

ibase_num_fields -- Get the number of fields in a result set

Description

int **ibase_num_fields** (resource result_id)

Returns an integer containing the number of fields in a result set.

```
<?php
    $rs = ibase_query("SELECT * FROM tablename");
    $coln = ibase_num_fields($rs);
    for ($i = 0; $i < $coln; $i++) {
        $col_info = ibase_field_info($rs, $i);
        echo "name: " . $col_info['name'] . "\n";
        echo "alias: " . $col_info['alias'] . "\n";
        echo "relation: " . $col_info['relation'] . "\n";
        echo "length: " . $col_info['length'] . "\n";
        echo "type: " . $col_info['type'] . "\n";
    }
?>
```

See also: **ibase_field_info()**.

ibase_num_params

(PHP 5)

ibase_num_params -- Return the number of parameters in a prepared query

Description

int **ibase_num_params** (resource query)

This function returns the number of parameters in the prepared query specified by *query*. This is the number of binding arguments that must be present when calling **ibase_execute()**.

See also **ibase_prepare()** and **ibase_param_info()**.

ibase_param_info

(PHP 5)

ibase_param_info -- Return information about a parameter in a prepared query

Description

array **ibase_param_info** (resource query, int param_number)

Returns an array with information about a parameter after a query has been prepared. The array is in the form of name, alias, relation, length, type.

See also **ibase_field_info()** and **ibase_num_params()**.

ibase_pconnect

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

ibase_pconnect --

Descripcion

ibase_pconnect ( )

ibase_prepare

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

ibase_prepare --

Descripcion

ibase_prepare ( )

ibase_query

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

ibase_query --

Descripcion

ibase_query ( )

ibase_restore

(PHP 5)

ibase_restore -- Initiates a restore task in the service manager and returns immediately

Description

mixed **ibase_restore** (resource service_handle, string source_file, string dest_db [, int options [, bool verbose]])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ibase_rollback_ret

(PHP 5)

ibase_rollback_ret -- Roll back a transaction without closing it

Description

bool **ibase_rollback_ret** ([resource link_or_trans_identifier])

If called without an argument, this function rolls back the default transaction of the default link. If the argument is a connection identifier, the default transaction of the corresponding connection will be rolled back. If the argument is a transaction identifier, the corresponding transaction will be rolled back. The transaction context will be retained, so statements executed from within this transaction will not be invalidated. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

ibase_rollback

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

ibase_rollback -- Roll back a transaction

Description

bool **ibase_rollback** ([resource link_or_trans_identifier])

If called without an argument, this function rolls back the default transaction of the default link. If the argument is a connection identifier, the default transaction of the corresponding connection will be rolled back. If the argument is a transaction identifier, the corresponding transaction will be rolled back. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

ibase_server_info

(PHP 5)

ibase_server_info -- Request information about a database server

Description

string **ibase_server_info** (resource service_handle, int action)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ibase_service_attach

(PHP 5)

ibase_service_attach -- Connect to the service manager

Description

resource **ibase_service_attach** (string host, string dba_username, string dba_password)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ibase_service_detach

(PHP 5)

ibase_service_detach -- Disconnect from the service manager

Description

bool **ibase_service_detach** (resource service_handle)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ibase_set_event_handler

(PHP 5)

ibase_set_event_handler -- Register a callback function to be called when events are posted

Description

resource **ibase_set_event_handler** (callback event_handler, string event_name1 [, string event_name2 [, string ...]])

resource **ibase_set_event_handler** (resource connection, callback event_handler, string event_name1 [, string event_name2 [, string ...]])

This function registers a PHP user function as event handler for the specified events. The callback is called with the event name and the link resource as arguments whenever one of the specified events is posted by the database. The callback must return **FALSE** if the event handler should be canceled. Any other return value is ignored. This function accepts up to 15 event arguments.

```
<?php

function event_handler($event_name, $link)
{
    if ($event_name=="NEW ORDER") {
        // process new order
        ibase_query($link, "UPDATE orders SET status='handled'");
    } else if ($event_name=="DB_SHUTDOWN") {
        // free event handler
        return false;
    }
}

ibase_set_event_handler($link, "event_handler", "NEW_ORDER", "DB_SHUTDOWN");
?>
```

The return value is an event resource. This resource can be used to free the event handler using **ibase_free_event_handler()**.

See also **ibase_free_event_handler()** and **ibase_wait_event()**.

ibase_timefmt

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

ibase_timefmt --

Descripcion

ibase_timefmt ( )

ibase_trans

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

ibase_trans -- Begin a transaction

Description

resource **ibase_trans** ([int trans_args [, resource link_identifier]])

Begins a transaction.

trans_args can be a combination of IBASE_READ, IBASE_WRITE, IBASE_COMMITTED, IBASE_CONSISTENCY, IBASE_CONCURRENCY, IBASE_REC_VERSION, IBASE_REC_NO_VERSION, IBASE_WAIT and IBASE_NOWAIT.

> **Nota:** The behaviour of this function has been changed in PHP 5.0.0. The first call to **ibase_trans()** will not return the default transaction of a connection. All transactions started by **ibase_trans()** will be rolled back at the end of the script if they were not committed or rolled back by either **ibase_commit()** or **ibase_rollback()**.
>
> **Nota:** In PHP 5.0.0. and up, this function will accept multiple *trans_args* and *link_identifier* arguments. This allows transactions over multiple database connections, which are committed using a 2-phase commit algorithm. This means you can rely on the updates to either succeed in every database, or fail in every database. It does NOT mean you can use tables from different databases in the same query!
>
> If you use transactions over multiple databases, you will have to specify both the *link_id* and *transaction_id* in calls to **ibase_query()** and **ibase_prepare()**.

ibase_wait_event

(PHP 5)

ibase_wait_event -- Wait for an event to be posted by the database

Description

string **ibase_wait_event** (string event_name1 [, string event_name2 [, string ...]])

string **ibase_wait_event** (resource connection, string event_name1 [, string event_name2 [, string ...]])

This function suspends execution of the script until one of the specified events is posted by the database. The name of the event that was posted is returned. This function accepts up to 15 event arguments.

See also **ibase_set_event_handler()** and **ibase_free_event_handler()**.

XLVII. Funciones ICAP [obsoletas]

Introducción

> **Nota:** Icap será eliminado en el futuro cercano. Ni este módulo, ni las versiones de la biblioteca icap siguen siendo soportadas. Si desea usar funcionalidad de calendario en PHP, use mcal en su lugar.

Requirimientos

La biblioteca icap tiene que estar instalada, la cual no es soportada ni se encuentra disponible ahora.

Instalación

Para hacer que estas funciones trabajen, necesita compilar PHP con *--with-icap*.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Esta extensión no tiene ninguna constante definida.

Tabla de contenidos

icap_close

(no version information, might be only in CVS)

icap_close -- Cierra un stream ICAP

Descripción

int **icap_close** (int stream_icap, int banderas)

Cierra el stream ICAP dado.

icap_create_calendar

(PHP 4 <= 4.2.3)

icap_create_calendar -- Create a new calendar

Description

string **icap_create_calendar** (int stream_id, string calendar)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

icap_delete_calendar

(PHP 4 <= 4.2.3)

icap_delete_calendar -- Delete a calendar

Description

string **icap_delete_calendar** (int stream_id, string calendar)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

icap_delete_event

(PHP 4 <= 4.2.3)

icap_delete_event -- Borra un evento de un calendario ICAP

Descripción

int **icap_delete_event** (int id_evento)

icap_delete_event() borra el evento de calendario especificado por el *id_evento*.

Devuelve **`TRUE`**.

icap_fetch_event

(PHP 4 <= 4.2.3)

icap_fetch_event -- Obtiene un evento del stream de calendario/

Descripción

object **icap_fetch_event** (stream stream_icap, id id_evento, options opciones)

icap_fetch_event() obtiene el evento del stream de calendario especificado por el parámetro *id_evento*.

Devuelve un objeto de evento compuesto por:

- int id - ID de dicho evento.
- int public - **TRUE** si el evento es público, **FALSE** si es privado.
- string category - Cadena con la categoría del evento.
- string title - Cadena de título del evento.
- string description - Cadena de descripción del evento.
- int alarm - Número de minutos antes que el evento envíe una alarma/recordatorio.
- object start - Objeto que contiene una entrada datetime (fecha/hora).
- object end - Objeto que contiene una entrada datetime.

Todas las entradas datetime consisten en un objeto compuesto por:

- int year - año
- int month - mes
- int mday - día del mes
- int hour - hora
- int min - minutos
- int sec - segundos

icap_list_alarms

(PHP 4 <= 4.2.3)

icap_list_alarms -- Devuelve una lista de los eventos que una alarma ha disparado en el instante dado

Descripción

array **icap_list_alarms** (stream stream_icap, datetime instante_alarma)

Devuelve una tabla de identificadores de evento para los que una alarma debiera apagarse en el instante indicado.

La función **icap_list_alarms()** toma una estructura datetime para un stream de calendario. Se devuelve una tabla de los identificadores de evento de todas las alarmas que debieran apagarse en el instante indicado.

Todas las entradas datetime consisten en un objeto compuesto por:

- int year - año
- int month - mes
- int mday - día del mes
- int hour - hora
- int min - minutos
- int sec - segundos

icap_list_events

(PHP 4 <= 4.2.3)

icap_list_events -- Devuelve una lista de eventos entre dos instantes dados

Descripción

array **icap_list_events** (stream stream_icap, datetime instante_inicio, datetime instante_final)

Devuelve una tabla de ID de evento que están entre los dos instantes dados.

La función **icap_list_events()** toma un instante de inicio y uno de final para un stream de calendario. Se devuelve una tabla de ID de evento que están entre los instantes dados.

Todas las entradas datetime consisten en un objeto compuesto por:

- int year - año
- int month - mes
- int mday - día del mes
- int hour - hora
- int min - minutos

- int sec - segundos

icap_open

(PHP 4 <= 4.2.3)

icap_open -- Abre una conexión ICAP

Descripción

stream **icap_open** (string calendario, string usuario, string clave, string opciones)

Si hay éxito devuelve un stream (flujo) ICAP, o **`FALSE`** en caso de error.

icap_open() abre una conexión ICAP con el servidor de *calendario* especificado. Si se especifica el parámetro opcional *opciones*, también éste es pasado a dicho buzón.

icap_rename_calendar

(PHP 4 <= 4.2.3)

icap_rename_calendar -- Rename a calendar

Description

string **icap_rename_calendar** (int stream_id, string old_name, string new_name)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

icap_reopen

(PHP 4 <= 4.2.3)

icap_reopen -- Reopen ICAP stream to new calendar

Description

int **icap_reopen** (int stream_id, string calendar [, int options])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

icap_snooze

(PHP 4 <= 4.2.3)

icap_snooze -- Apaga la alarma de un evento

Descripción

int **icap_snooze** (int id_evento)

icap_snooze() apaga la alarma del evento de calendario especificado por el *id_evento*.

Returns **`TRUE`**.

icap_store_event

(PHP 4 <= 4.2.3)

icap_store_event -- Almacena un evento en un calendario ICAP

Descripción

int **icap_store_event** (int stream_icap, object evento)

icap_store_event() Guarda un evento en un calendario ICAP. Un objeto de evento consiste en:

- int public - 1 si es público, 0 si es privado;
- string category - Cadena con la categoría del evento.
- string title - Cadena de título del evento.
- string description - Cadena de descripción del evento.
- int alarm - Número de minutos antes que el evento envíe una alarma/recordatorio.
- object start - Objeto que contiene una entrada datetime (fecha/hora).
- object end - Objeto que contiene una entrada datetime.

Todas las entradas datetime consisten en un objeto compuesto por:

- int year - año
- int month - mes
- int mday - día del mes
- int hour - hora
- int min - minutos

- int sec - segundos

Devuelve **TRUE** en caso de éxito y **FALSE** en caso de error.

XLVIII. Funciones iconv

Introducción

Este módulo contiene una interfaz con la facilidad de conversión de juegos de caracteres iconv. Con éste módulo, es posible convertir una cadena representada por un juego de caracteres local a una representada por otro juego de caracteres, que puede ser el juego de caracteres Unicode. Los juegos de caracteres soportados dependen de la implementación de iconv en su sistema. Note que la función iconv en algunos sistemas puede que no funcione como lo espera. En tal caso, sería una buena idea instalar la biblioteca libiconv GNU. Es muy probable que consiga unos resultados más consistentes.

A partir de PHP 5.0.0, esta extensión viene con varias funciones utilitarias que le ayudan a escribir scripts multi-lingües. Echemos un vistazo a las siguientes secciones para explorar las nuevas características.

Requirimientos

No necesita de nada si el sistema que usa es uno de los sistemas recientes compatibles con POSIX, ya que las bibliotecas C estándar que vienen con ellos ofrecen la facilidad iconv. De otro modo, es necesario instalar la biblioteca libiconv en su sistema.

Instalación

Para usar las funciones de éste módulo, el binario PHP debe ser compilado con la siguiente línea de configuración: *--with-iconv[=DIR]*.

> **Nota para Usuarios de WindowsÃ‚Â®:** Para habilitar éste módulo en un entorno WindowsÃ‚Â®, necesita colocar un archivo DLL llamado *iconv.dll* o *iconv-1.3.dll* (antes de 4.2.1), el cual hace parte del paquete binario PHP/Win32, en un directorio especificado por la variable de entorno *PATH* o uno de los directorios de sistema de su instalación de WindowsÃ‚Â®.

Configuración en tiempo de ejecución

El comportamiento de estas funciones está afectado por los valores definidos en `php.ini`.

Tabla 1. Opciones de configuración de iconv

Nombre	Predeterminado	Modificable
iconv.input_encoding	ICONV_INPUT_ENCODING	PHP_INI_ALL

Nombre	Predeterminado	Modificable
iconv.output_encoding	ICONV_OUTPUT_ENCODING	PHP_INI_ALL
iconv.internal_encoding	ICONV_INTERNAL_ENCODING	PHP_INI_ALL

Para más detalles sobre las constantes PHP_INI_* y su definición, vea **ini_set()**.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Desde PHP 4.3.0, es posible identificar, en tiempo de ejecución, qué implementación de *iconv* es adoptada por esta extensión.

Tabla 2. Constantes de *iconv*

Nombre	Tipo	Descripción
ICONV_IMPL	**string**	El nombre de la implementación
ICONV_VERSION	**string**	La versión de la implementación

Nota: Escribir scripts que dependan de la implementación con éstas constantes no se recomienda en absoluto.

A partir de PHP 5.0.0, las siguientes constantes se encuentran disponibles también:

Tabla 3. Constantes de *iconv* disponibles desde PHP 5.0.0

Nombre	Tipo	Descripción
ICONV_MIME_DECODE_STRICT	**integer**	Una máscara de bits usada para **iconv_mime_decode()**
ICONV_MIME_DECODE_CONTINUE_ON_ERROR	**integer**	Una máscara de bits usada para **iconv_mime_decode()**

Ver también

Vea también las funciones GNU Recode.

Tabla de contenidos

iconv_strlen -- Returns the character count of string
iconv_strpos -- Finds position of first occurrence of a needle within a haystack
iconv_strrpos -- Finds the last occurrence of a needle within the specified range of haystack
iconv_substr -- Cut out part of a string
iconv -- Convert string to requested character encoding
ob_iconv_handler -- Convert character encoding as output buffer handler

iconv_get_encoding

(PHP 4 >= 4.0.5, PHP 5)

iconv_get_encoding -- Retrieve internal configuration variables of iconv extension

Description

mixed **iconv_get_encoding** ([string type])

iconv_get_encoding() returns the current value of the internal configuration variable if successful, or **FALSE** on failure.

The value of the optional *type* can be:

all
input_encoding
output_encoding
internal_encoding

If *type* is omitted or set to "all", **iconv_get_encoding()** returns an array that stores all these variables.

Ejemplo 1. iconv_get_encoding() example

```
<pre>
<?php
iconv_set_encoding("internal_encoding", "UTF-8");
iconv_set_encoding("output_encoding", "ISO-8859-1");
var_dump(iconv_get_encoding('all'));
?>
</pre>
```

The printout of the above program will be:

```
Array
(
    [input_encoding] => ISO-8859-1
    [output_encoding] => ISO-8859-1
    [internal_encoding] => UTF-8
)
```

See also **iconv_set_encoding()** and **ob_iconv_handler()**.

iconv_mime_decode_headers

(PHP 5)

iconv_mime_decode_headers -- Decodes multiple *MIME* header fields at once

Description

array **iconv_mime_decode_headers** (string encoded_headers [, int mode [, string charset]])

Returns an associative array that holds a whole set of *MIME* header fields specified by *encoded_headers* on success, or **FALSE** if an error occurs during the decoding.

Each key of the return value represents an individual field name and the corresponding element represents a field value. If more than one field of the same name are present, **iconv_mime_decode_headers()** automatically incorporates them into a numerically indexed array in the order of occurrence.

mode determines the behaviour in the event **iconv_mime_decode_headers()** encounters a malformed *MIME* header field. You can specify any combination of the following bitmasks.

Tabla 1. Bitmasks acceptable to iconv_mime_decode_headers()

Value	Constant	Description
1	ICONV_MIME_DECODE_STRICT	If set, the given header is decoded in full conformance with the standards defined in RFC2047. This option is disabled by default because there are a lot of broken mail user agents that don't follow the specification and don't produce correct *MIME* headers.
2	ICONV_MIME_DECODE_CONTINUE_ON_ERROR	If set, **iconv_mime_decode_headers()** attempts to ignore any grammatical errors and continue to process a given header.

The optional *charset* parameter specifies the character set to represent the result by. If omitted, iconv.internal_charset will be used.

Ejemplo 1. iconv_mime_decode_headers() example

```
<?php
$headers_string = <<<EOF
Subject: =?UTF-8?B?UHLDvGZ1bmcgUHLDvGZ1bmc=?=
To: example@example.com
Date: Thu, 1 Jan 1970 00:00:00 +0000
Message-Id: <example@example.com>
Received: from localhost (localhost [127.0.0.1]) by localhost
    with SMTP id example for <example@example.com>
    Thu, 1 Jan 1970 00:00:00 +0000 (UTC)
    (envelope-from example-return-0000-example=example.com@example.com)
Received: (qmail 0 invoked by uid 65534); 1 Thu 2003 00:00:00 +0000

EOF;

$headers =  iconv_mime_decode_headers($headers_string, 0, "ISO-8859-1");
print_r($headers);
?>
```

El resultado del ejemplo seria:

```
Array
(
    [Subject] => Prüfung Prüfung
    [To] => example@example.com
    [Date] => Thu, 1 Jan 1970 00:00:00 +0000
    [Message-Id] => <example@example.com>
    [Received] => Array
        (
            [0] => from localhost (localhost [127.0.0.1]) by localhost with SMTP id exa
            [1] => (qmail 0 invoked by uid 65534); 1 Thu 2003 00:00:00 +0000
        )

)
```

See also **iconv_mime_decode()**, **mb_decode_mimeheader()**, **imap_mime_header_decode()**, **imap_base64()** and **imap_qprint()**.

iconv_mime_decode

(PHP 5)

iconv_mime_decode -- Decodes a *MIME* header field

Description

string **iconv_mime_decode** (string encoded_header [, int mode [, string charset]])

Returns a decoded *MIME* field on success, or **FALSE** if an error occurs during the decoding.

mode determines the behaviour in the event **iconv_mime_decode()** encounters a malformed *MIME* header field. You can specify any combination of the following bitmasks.

Tabla 1. Bitmasks acceptable to iconv_mime_decode()

Value	Constant	Description
1	ICONV_MIME_DECODE_STRICT	If set, the given header is decoded in full conformance with the standards defined in RFC2047. This option is disabled by default because there are a lot of broken mail user agents that don't follow the specification and don't produce correct *MIME* headers.
2	ICONV_MIME_DECODE_CONTINUE_ON_ERROR	If set, **iconv_mime_decode()** attempts to continue to process the given header even though an error occurs.

The optional *charset* parameter specifies the character set to represent the result by. If omitted, iconv.internal_charset will be used.

Ejemplo 1. iconv_mime_decode() example

```
<?php
// This yields "Subject: Prüfung Prüfung"
echo iconv_mime_decode("Subject: =?UTF-8?B?UHLDvGZ1bmcgUHLDvGZ1bmc=?=",
                       0, "ISO-8859-1");
?>
```

See also **iconv_mime_decode_headers()**, **mb_decode_mimeheader()**, **imap_mime_header_decode()**, **imap_base64()** and **imap_qprint()**.

iconv_mime_encode

(PHP 5)

iconv_mime_encode -- Composes a *MIME* header field

Description

string **iconv_mime_encode** (string field_name, string field_value [, array preferences])

Composes and returns a string that represents a valid *MIME* header field, which looks like the following:

```
Subject: =?ISO-8859-1?Q?Pr=FCfung_f=FCr?= Entwerfen von einer MIME kopfzeile
```

In the above example, "Subject" is the field name and the portion that begins with "=?ISO-8859-1?..." is the field value.

You can control the behaviour of **iconv_mime_encode()** by specifying an associative array that contains configuration items to the optional third parameter *preferences*. The items supported by **iconv_mime_encode()** are listed below. Note that item names are treated case-sensitive.

Tabla 1. Configuration items supported by iconv_mime_encode()

Item	Type	Description	Default value	Example
scheme	**boolean**	Specifies the method to encode a field value by. The value of this item may be either "B" or "Q", where "B" stands for *base64* encoding scheme and "Q" stands for *quoted-printable* encoding scheme.	B	B
input-charset	**string**	Specifies the character set in which the first parameter *field_name* and the second parameter *field_value* are presented. If not given, **iconv_mime_encode()** assumes those parameters are presented to it in the iconv.internal_charset ini setting.	iconv.internal_charset	ISO-8859-1
output-charset	**string**	Specifies the character set to use to compose the *MIME* header. If not given, the same value as *input-charset* will be used.	the same value as *input-charset*	UTF-8
line-length	**integer**	Specifies the maximum length of the header lines. The resulting header is "folded" to a set of multiple lines in case the resulting header field would be longer than the value of this parameter, according to RFC2822 - Internet Message Format. If not given, the length will be limited to 76 characters.	76	996
line-break-chars	**string**	Specifies the sequence of characters to append to each line as an end-of-line sign when "folding" is performed on a long header field. If not given, this defaults to "\r\n" (*CR LF*). Note that this parameter is always treated as an ASCII string regardless of the value of *input-charset*.	\r\n	\n

Ejemplo 1. iconv_mime_encode() example:

```
<?php
$preferences = array(
    "input-charset" => "ISO-8859-1",
    "output-charset" => "UTF-8",
    "line-length" => 76,
    "line-break-chars" => "\n"
);
$preferences["scheme"] = "Q";
// This yields "Subject: =?UTF-8?Q?Pr=C3=BCfung_Pr=C3=BCfung?="
echo iconv_mime_encode("Subject", "Prüfung Prüfung", $preferences);

$preferences["scheme"] = "B";
// This yields "Subject: =?UTF-8?B?UHLDvGZ1bmcgUHLDvGZ1bmc=?="
echo iconv_mime_encode("Subject", "Prüfung Prüfung", $preferences);
?>
```

See also **imap_binary()**, **mb_encode_mimeheader()** and **imap_8bit()**.

iconv_set_encoding

(PHP 4 >= 4.0.5, PHP 5)

iconv_set_encoding -- Set current setting for character encoding conversion

Description

bool **iconv_set_encoding** (string type, string charset)

iconv_set_encoding() changes the value of the internal configuration variable specified by *type* to *charset*. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

The value of *type* can be any one of those:

input_encoding
output_encoding
internal_encoding

Ejemplo 1. iconv_set_encoding() example:

```
<?php
iconv_set_encoding("internal_encoding", "UTF-8");
iconv_set_encoding("output_encoding", "ISO-8859-1");
?>
```

See also **iconv_get_encoding()** and **ob_iconv_handler()**.

iconv_strlen

(PHP 5)

iconv_strlen -- Returns the character count of string

Description

int **iconv_strlen** (string str [, string charset])

Returns the character count of *str*.

In contrast to **strlen()**, **iconv_strlen()** counts the occurrences of characters in the given byte sequence *str* on the basis of the specified character set, the result of which is not necessarily identical to the length of the string in byte.

If *charset* parameter is omitted, *str* is assumed to be encoded in iconv.internal_charset.

See also **strlen()** and **mb_strlen()**.

iconv_strpos

(PHP 5)

iconv_strpos -- Finds position of first occurrence of a needle within a haystack

Description

int **iconv_strpos** (string haystack, string needle [, int offset [, string charset]])

Returns the numeric position of the first occurrence of *needle* in *haystack*.

The optional *offset* parameter specifies the position from which the search should be performed.

If *needle* is not found, **iconv_strpos()** will return **FALSE**.

Aviso
Esta función puede devolver **FALSE**, pero también puede devolver un valor no-booleano que será evaluado **FALSE**, como por ejemplo *0* o "". Por favor, lea la sección Booleans para más información. Utilice el operador === para comprobar el valor devuelto por esta función.

If *haystack* or *needle* is not a string, it is converted to a string and applied as the ordinal value of a character.

In contrast to **strpos()**, the return value of **iconv_strpos()** is the number of characters that appear before the needle, rather than the offset in bytes to the position where the needle has been found. The characters are counted on the basis of the specified character set *charset*.

If *charset* parameter is omitted, *string* are assumed to be encoded in iconv.internal_charset.

See also **strpos()**, **iconv_strrpos()** and **mb_strpos()**.

iconv_strrpos

(PHP 5)

iconv_strrpos -- Finds the last occurrence of a needle within the specified range of haystack

Description

string **iconv_strrpos** (string haystack, string needle [, string charset])

Returns the numeric position of the last occurrence of *needle* in *haystack*.

If *needle* is not found, **iconv_strrpos()** will return **FALSE**.

Aviso
Esta función puede devolver **FALSE**, pero también puede devolver un valor no-booleano que será evaluado **FALSE**, como por ejemplo *0* o "". Por favor, lea la sección Booleans para más información. Utilice el operador === para comprobar el valor devuelto por esta función.

If *haystack* or *needle* is not a string, it is converted to a string and applied as the ordinal value of a character.

In contrast to **strpos()**, the return value of **iconv_strrpos()** is the number of characters that appear before the needle, rather than the offset in bytes to the position where the needle has been found. The characters are counted on the basis of the specified character set *charset*.

See also **strrpos()**, **iconv_strpos()** and **mb_strrpos()**.

iconv_substr

(PHP 5)

iconv_substr -- Cut out part of a string

Description

string **iconv_substr** (string str, int offset [, int length [, string charset]])

Returns the portion of *str* specified by the *start* and *length* parameters.

If *start* is non-negative, **iconv_substr()** cuts the portion out of *str* beginning at *start*'th character, counting from zero.

If *start* is negative, **iconv_substr()** cuts out the portion beginning at the position, *start* characters away from the end of *str*.

If *length* is given and is positive, the return value will contain at most *length* characters of the portion that begins at *start* (depending on the length of *string*). If *str* is shorter than *start* characters long, **FALSE** will be returned.

If negative *length* is passed, **iconv_substr()** cuts the portion out of *str* from the *start*'th character up to the character that is *length* characters away from the end of the string. In case *start* is also negative, the start position is calculated beforehand according to the rule explained above.

Note that *offset* and *length* parameters are always deemed to represent offsets that are calculated on the basis of the character set determined by *charset*, whilst the counterpart **substr()** always takes these for byte offsets. If *charset* is not given, the character set is determined by the iconv.internal_encoding ini setting.

See also **substr()**, **mb_substr()** and **mb_strcut()**.

iconv

(PHP 4 >= 4.0.5, PHP 5)

iconv -- Convert string to requested character encoding

Description

string **iconv** (string in_charset, string out_charset, string str)

Performs a character set conversion on the string *str* from *in_charset* to *out_charset*. Returns the converted string or **FALSE** on failure.

If you append the string *//TRANSLIT* to *out_charset* transliteration is activated. This means that when a character can't be represented in the target charset, it can be approximated through one or several similarly looking characters. If you append the string *//IGNORE*, characters that cannot be represented in the target charset are silently discarded. Otherwise, *str* is cut from the first illegal character.

Ejemplo 1. iconv() example:

```
<?php
echo iconv("ISO-8859-1", "UTF-8", "This is a test.");
?>
```

ob_iconv_handler

(PHP 4 >= 4.0.5, PHP 5)

ob_iconv_handler -- Convert character encoding as output buffer handler

Description

array **ob_iconv_handler** (string contents, int status)

It converts the string encoded in *internal_encoding* to *output_encoding*.

internal_encoding and *output_encoding* should be defined by **iconv_set_encoding()** or in the configuration file `php.ini`.

Ejemplo 1. ob_iconv_handler() example:

```
<?php
ob_start("ob_iconv_handler"); // start output buffering
?>
```

See also **iconv_get_encoding()**, **iconv_set_encoding()** and output-control functions.

XLIX. ID3 Functions

Introducción

These functions let you read and manipulate ID3 tags. ID3 tags are used in MP3 files to store title of

the song, as well as information about the artist, album, genre, year and track number.

Since version 0.2 it is also possible to extract text frames from ID3 v2.2+ tags.

Requirimientos

No se necesitan bibliotecas externas para construir esta extensión

---

Instalación

id3 is part of PECL and can be installed using the PEAR installer. To compile PHP with id3 support, download the sourcecode, put it in php-src/ext/id3 and compile PHP using --enable-id3.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Predefined Constants

Most of the id3 functions either let you specify or return a tag version. In order to specify the version please use on of these constants.

`ID3_V1_0` (**integer**)

> **`ID3_V1_0`** is used if you are working with ID3 V1.0 tags. These tags may contain the fields title, artist, album, genre, year and comment.

`ID3_V1_1` (**integer**)

> **`ID3_V1_1`** is used if you are working with ID3 V1.1 tags. These tags may all information contained in v1.0 tags plus the track number.

`ID3_V2_1` (**integer**)

> **`ID3_V2_1`** is used if you are working with ID3 V2.1 tags.

`ID3_V2_2` (**integer**)

> **`ID3_V2_2`** is used if you are working with ID3 V2.2 tags.

ID3_V2_3 (integer)

ID3_V2_3 is used if you are working with ID3 V2.3 tags.

ID3_V2_4 (integer)

ID3_V2_4 is used if you are working with ID3 V2.4 tags.

ID3_BEST (integer)

ID3_BEST is used if would like to let the id3 functions determine which tag version should be used.

Tabla de contenidos

id3_get_frame_long_name

(no version information, might be only in CVS)

id3_get_frame_long_name -- Get the long name of an ID3v2 frame

Description

string **id3_get_frame_long_name** (string frameId)

id3_get_frame_long_name() returns the long name for an ID3v2 frame.

Ejemplo 1. id3_get_frame_long_name() example

```
<?php
$longName = id3_get_frame_long_name("TOLY");
echo $longName;
?>
```

This will output:

```
Original lyricist(s)/text writer(s)
```

See also **id3_get_frame_short_name()**.

id3_get_frame_short_name

(no version information, might be only in CVS)

id3_get_frame_short_name -- Get the short name of an ID3v2 frame

Description

string **id3_get_frame_short_name** (string frameId)

id3_get_frame_short_name() returns the short name for an ID3v2 frame.

Ejemplo 1. id3_get_frame_short_name() example

```
<?php
$shortName = id3_get_frame_short_name("TOLY");
echo $shortName;
?>
```

This will output:

```
originalLyricist
```

The values returned by **id3_get_short_name()** are used in the array returned by **id3_get_tag()**.

See also **id3_get_frame_long_name()**.

id3_get_genre_id

(no version information, might be only in CVS)

id3_get_genre_id -- Get the id for a genre

Description

int **id3_get_genre_id** (string genre)

id3_get_genre_id() returns the id for a genre. If the specified genre is not available in the genre list, **id3_get_genre_id()** will return **FALSE**

In an ID3 tag, the genre is stored using a integer ranging from 0 to 147.

Ejemplo 1. id3_get_genre_id() example

```
<?php
$id = id3_get_genre_id("Alternative");
echo $id;
?>
```

This will output:

```
20
```

See also **id3_get_genre_list()** and **id3_get_genre_name()**.

id3_get_genre_list

(no version information, might be only in CVS)

id3_get_genre_list -- Get all possible genre values

Description

array **id3_get_genre_list** (void)

id3_get_genre_list() returns an array containing all possible genres that may be stored in an ID3 tag. This list has been created by Eric Kemp and later extended by WinAmp.

This function is useful to provide you users a list of genres from which they may choose one. When updating the ID3 tag you will always have to specify the genre as an integer ranging from 0 to 147.

Ejemplo 1. id3_get_genre_list() example

```
<?php
$genres = id3_get_genre_list();
print_r($genres);
?>
```

This will output:

```
Array
(
    [0] => Blues
    [1] => Classic Rock
    [2] => Country
    [3] => Dance
    [4] => Disco
    [5] => Funk
    [6] => Grunge
    [7] => Hip-Hop
    [8] => Jazz
    [9] => Metal
    [10] => New Age
    [11] => Oldies
    [12] => Other
    [13] => Pop
    [14] => R&B
    [15] => Rap
    [16] => Reggae
    [17] => Rock
    [18] => Techno
    [19] => Industrial
    [20] => Alternative
    [21] => Ska
    [22] => Death Metal
    [23] => Pranks
    [24] => Soundtrack
    [25] => Euro-Techno
    [26] => Ambient
    [27] => Trip-Hop
    [28] => Vocal
    [29] => Jazz+Funk
    [30] => Fusion
    [31] => Trance
    [32] => Classical
    [33] => Instrumental
    [34] => Acid
    [35] => House
    [36] => Game
    [37] => Sound Clip
    [38] => Gospel
    [39] => Noise
    [40] => Alternative Rock
    [41] => Bass
    [42] => Soul
    [43] => Punk
    [44] => Space
    [45] => Meditative
    [46] => Instrumental Pop
    [47] => Instrumental Rock
    [48] => Ethnic
    [49] => Gothic
    [50] => Darkwave
    [51] => Techno-Industrial
    [52] => Electronic
    [53] => Pop-Folk
    [54] => Eurodance
    [55] => Dream
    [56] => Southern Rock
    [57] => Comedy
    [58] => Cult
    [59] => Gangsta
    [60] => Top 40
    [61] => Christian Rap
    [62] => Pop/Funk
    [63] => Jungle
    [64] => Native US
    [65] => Cabaret
    [66] => New Wave
    [67] => Psychadelic
    [68] => Rave
    [69] => Showtunes
```

See also **id3_get_genre_name()** and **id3_get_genre_id()**.

id3_get_genre_name

(no version information, might be only in CVS)

id3_get_genre_name -- Get the name for a genre id

Description

string **id3_get_genre_name** (int genre_id)

id3_get_genre_name() returns the name for a genre id.

In an ID3 tag, the genre is stored using a integer ranging from 0 to 147.

Ejemplo 1. id3_get_genre_name() example

```
<?php
$genre = id3_get_genre_name(20);
echo $genre;
?>
```

This will output:

```
Alternative
```

See also **id3_get_genre_list()** and **id3_get_genre_id()**.

id3_get_tag

(no version information, might be only in CVS)

id3_get_tag -- Get all information stored in an ID3 tag

Description

array **id3_get_tag** (string filename [, int version])

id3_get_tag() is used to get all information stored in the id3 tag of the specified file.

> **Nota:** Instead of a filename you may also pass a valid stream resource.

The optional *version* parameter allows you to specify the version of the tag as MP3 files may contain both, version 1.x and version 2.x tags.

Ejemplo 1. id3_get_tag() example

```
<?php
$tag = id3_get_tag( "path/to/example.mp3" );
print_r($tag);
?>
```

This will output something like:

```
Array
(
    [title] => DN-38416
    [artist] => Re:\Legion
    [album] => Reflections
    [year] => 2004
    [genre] => 19
)
```

The key *genre* will contain an integer between 0 and 147. You may use **id3_get_genre_name()** to convert it to a human readable string.

Since version 0.2 **id3_get_tag()** also supports ID3 tags of version 2.2, 2.3 and 2.4. To extract information from these tags, pass one of the constants ID3_V2_2, ID3_V2_3 or ID3_V2_4 as the second parameter.

Ejemplo 2. id3_get_tag() example

```
<?php
$tag = id3_get_tag( "path/to/example2.mp3", ID3_V2_3 );
print_r($tag);
?>
```

This will output something like:

```
Array
(
    [copyright] => Dirty Mac
    [originalArtist] => Dirty Mac
    [composer] => Marcus GÃƒÂ¶tze
    [artist] => Dirty Mac
    [title] => Little Big Man
    [album] => Demo-Tape
    [track] => 5/12
    [genre] => (17)Rock
    [year] => 2001
)
```

ID3 v2.x tags can contain a lot more information about the MP3 file than ID3 v1.x tags.

See also **id3_set_tag()**, **id3_remove_tag()** and **id3_get_version()**.

id3_get_version

(no version information, might be only in CVS)

id3_get_version -- Get version of an ID3 tag

Description

int **id3_get_version** (string filename)

id3_get_version() retrieves the version(s) of the ID3 tag(s) in the MP3 file. As a tag can contain ID3 v1.x and v2.x tags, the return value of this function should be bitwise compared with the predefined constants **`ID3_V1_0`**, **`ID3_V1_1`** and **`ID3_V2`**.

> **Nota:** Instead of a filename you may also pass a valid stream resource.

Ejemplo 1. id3_get_version() example

```
<?php
$version = id3_get_version( "path/to/example.mp3" );
if ($version & ID3_V1_0) {
    echo "Contains a 1.x tag\n";
}
if ($version & ID3_V1_1) {
    echo "Contains a 1.1 tag\n";
}
if ($version & ID3_V2) {
    echo "Contains a 2.x tag\n";
}
?>
```

This will output something like:

```
Contains a 1.x tag
Contains a 1.1 tag
```

If a file contains an ID3 v1.1 tag, it always contains a 1.0 tag, as version 1.1 is just an extension of 1.0.

See also **id3_get_tag()**, **id3_set_tag()** and **id3_remove_tag()**.

id3_remove_tag

(no version information, might be only in CVS)

id3_remove_tag -- Remove an existing ID3 tag

Description

bool **id3_remove_tag** (string filename [, int version])

id3_remove_tag() is used to remove the information stored of an ID3 tag. If no tag has been present, it will return **FALSE** and leave the file as it was.

> **Nota:** Instead of a filename you may also pass a valid stream resource.

The optional *version* parameter allows you to specify the version of the tag as MP3 files may contain both, version 1.x and version 2.x tags.

Ejemplo 1. id3_remove_tag() example

```
<?php
$result = id3_remove_tag( "path/to/example.mp3", ID3_V1_0 );
if ($result === true) {
    echo "Tag succesfully removed\n";
}
?>
```

If the file is writable and contained a 1.0 tag, this will output:

```
Tag succesfully removed
```

> **Nota:** Currently **id3_remove_tag()** only supports version 1.0 and 1.1. If you choose to remove a 1.0 tag and the file contains a 1.1 tag, this tag will be removed, as v1.1 is only an extension of 1.0.

See also **id3_get_tag()**, **id3_set_tag()** and **id3_get_version()**.

id3_set_tag

(no version information, might be only in CVS)

id3_set_tag -- Update information stored in an ID3 tag

Description

bool **id3_set_tag** (string filename, array tag [, int version])

id3_set_tag() is used to change the information stored of an ID3 tag. If no tag has been present, it will be added to the file.

> **Nota:** Instead of a filename you may also pass a valid stream resource.

The optional *version* parameter allows you to specify the version of the tag as MP3 files may contain both, version 1.x and version 2.x tags.

Ejemplo 1. id3_set_tag() example

```
<?php
$data = array(
              "title" => "Re:Start",
              "artist" => "Re:\Legion",
              "comment" => "A nice track"
             );
$result = id3_set_tag( "path/to/example.mp3", $data, ID3_V1_0 );
if ($result === true) {
    echo "Tag succesfully updated\n";
}
?>
```

If the file is writable, this will output:

```
Tag succesfully updated
```

> **Nota:** Currently **id3_set_tag()** only supports version 1.0 and 1.1.

The following keys may be used in the associative array:

Tabla 1. Keys in the associative array

key	possible value	available in version
title	string with maximum of 30 characters	v1.0, v1.1
artist	string with maximum of 30 characters	v1.0, v1.1
album	string with maximum of 30 characters	v1.0, v1.1
year	4 digits	v1.0, v1.1
genre	integer value between 0 and 147	v1.0, v1.1
comment	string with maximum of 30 characters (28 in v1.1)	v1.0, v1.1
track	integer between 0 and 255	v1.1

See also **id3_get_tag()**, **id3_remove_tag()** and **id3_get_version()**.

L. Funciones de Informix

Introducción

El controlador de Informix para Informix (IDS) 7.x, SE 7.x, Universal Server (IUS) 9.x y IDS 2000 es implementado en "ifx.ec" y "php3_ifx.h" en el directorio de la extensión informix. El soporte para IDS 7.x es razonablemente completo, con soporte para columnas BYTE y TEXT. El soporte para IUS 9.x se encuentra parcialmente terminado: los nuevos tipos de datos se encuentran allí, pero el soporte para SLOB y CLOB aun está en construcción

---

Requirimientos

Notas de configuración: Necesita alguna versión de ESQL/C para compilar el controlador para Informix de PHP. Las versiones de ESQL/C a partir de 7.2x deben trabajar bien. ESQL/C es ahora parte del SDK de Cliente de Informix.

Asegúrese de que la variable "INFORMIXDIR" haya sido definida, y de que $INFORMIXDIR/bin se encuentre en su PATH antes de ejecutar el script "configure".

Instalación

To be able to use the functions defined in this module you must compile your PHP interpreter using the configure line *--with_informix[=DIR]*, where DIR is the Informix base install directory, defaults to nothing.

Configuración en tiempo de ejecución

El comportamiento de estas funciones está afectado por los valores definidos en `php.ini`.

Nota: Make sure that the Informix environment variables INFORMIXDIR and INFORMIXSERVER are available to the PHP ifx driver, and that the INFORMIX bin directory is in the PATH. Check this by running a script that contains a call to **phpinfo()** before you start testing. The **phpinfo()** output should list these environment variables. This is true for both CGI php and Apache mod_php. You may have to set these environment variables in your Apache startup script.

The Informix shared libraries should also be available to the loader (check LD_LIBRARY_PATH or ld.so.conf/ldconfig).

Some notes on the use of BLOBs (TEXT and BYTE columns): BLOBs are normally addressed by BLOB identifiers. Select queries return a "blob id" for every BYTE and TEXT column. You can get at the contents with "string_var = ifx_get_blob($blob_id);" if you choose to get the BLOBs in memory (with: "ifx_blobinfile(0);"). If you prefer to receive the content of BLOB columns in a file, use "ifx_blobinfile(1);", and "ifx_get_blob($blob_id);" will get you the filename. Use normal file I/O to get at the blob contents.

For insert/update queries you must create these "blob id's" yourself with "**ifx_create_blob()**;". You then plug the blob id's into an array, and replace the blob columns with a question mark (?) in the query string. For updates/inserts, you are responsible for setting the blob contents with **ifx_update_blob()**.

The behaviour of BLOB columns can be altered by configuration variables that also can be set at runtime:

configuration variable: ifx.textasvarchar

configuration variable: ifx.byteasvarchar

runtime functions:

ifx_textasvarchar(0): use blob id's for select queries with TEXT columns

ifx_byteasvarchar(0): use blob id's for select queries with BYTE columns

ifx_textasvarchar(1): return TEXT columns as if they were VARCHAR columns, so that you don't need to use blob id's for select queries.

ifx_byteasvarchar(1): return BYTE columns as if they were VARCHAR columns, so that you don't need to use blob id's for select queries.

configuration variable: ifx.blobinfile

runtime function:

ifx_blobinfile_mode(0): return BYTE columns in memory, the blob id lets you get at the contents.

ifx_blobinfile_mode(1): return BYTE columns in a file, the blob id lets you get at the file name.

If you set ifx_text/byteasvarchar to 1, you can use TEXT and BYTE columns in select queries just like normal (but rather long) VARCHAR fields. Since all strings are "counted" in PHP, this remains "binary safe". It is up to you to handle this correctly. The returned data can contain anything, you are responsible for the contents.

If you set ifx_blobinfile to 1, use the file name returned by ifx_get_blob(..) to get at the blob contents. Note that in this case YOU ARE RESPONSIBLE FOR DELETING THE TEMPORARY FILES CREATED BY INFORMIX when fetching the row. Every new row fetched will create new temporary files for every BYTE column.

The location of the temporary files can be influenced by the environment variable "blobdir", default is "." (the current directory). Something like: putenv (blobdir=tmpblob"); will ease the cleaning up of temp files accidentally left behind (their names all start with "blb").

Automatically trimming "char" (SQLCHAR and SQLNCHAR) data: This can be set with the configuration variable

ifx.charasvarchar: if set to 1 trailing spaces will be automatically trimmed, to save you some "chopping".

NULL values: The configuration variable ifx.nullformat (and the runtime function **ifx_nullformat()**) when set to **TRUE** will return **NULL** columns as the string "**NULL**", when set to **FALSE** they return the empty string. This allows you to discriminate between **NULL** columns and empty columns.

Tabla 1. Informix configuration options

Name	Default	Changeable
ifx.allow_persistent	"1"	PHP_INI_SYSTEM
ifx.max_persistent	"-1"	PHP_INI_SYSTEM
ifx.max_links	"-1"	PHP_INI_SYSTEM
ifx.default_host	NULL	PHP_INI_SYSTEM
ifx.default_user	NULL	PHP_INI_SYSTEM
ifx.default_password	NULL	PHP_INI_SYSTEM
ifx.blobinfile	"1"	PHP_INI_ALL
ifx.textasvarchar	"0"	PHP_INI_ALL
ifx.byteasvarchar	"0"	PHP_INI_ALL
ifx.charasvarchar	"0"	PHP_INI_ALL
ifx.nullformat	"0"	PHP_INI_ALL

For further details and definitions of the PHP_INI_* constants, see the Apéndice H.

A continuación se presenta una corta explicación de las directivas de configuración.

ifx.allow_persistent **boolean**

Whether to allow persistent Informix connections.

ifx.max_persistent **integer**

The maximum number of persistent Informix connections per process.

ifx.max_links **integer**

The maximum number of Informix connections per process, including persistent connections.

ifx.default_host **string**

The default host to connect to when no host is specified in **ifx_connect()** or **ifx_pconnect()**. Doesn't apply in safe mode.

ifx.default_user **string**

The default user id to use when none is specified in **ifx_connect()** or **ifx_pconnect()**. Doesn't apply in safe mode.

ifx.default_password **string**

The default password to use when none is specified in **ifx_connect()** or **ifx_pconnect()**.

Doesn't apply in safe mode.

ifx.blobinfile **boolean**

Set to **TRUE** if you want to return blob columns in a file, **FALSE** if you want them in memory. You can override the setting at runtime with **ifx_blobinfile_mode()**.

ifx.textasvarchar **boolean**

Set to **TRUE** if you want to return TEXT columns as normal strings in select statements, **FALSE** if you want to use blob id parameters. You can override the setting at runtime with **ifx_textasvarchar()**.

ifx.byteasvarchar **boolean**

Set to **TRUE** if you want to return BYTE columns as normal strings in select queries, **FALSE** if you want to use blob id parameters. You can override the setting at runtime with **ifx_textasvarchar()**.

ifx.charasvarchar **boolean**

Set to **TRUE** if you want to trim trailing spaces from CHAR columns when fetching them.

ifx.nullformat **boolean**

Set to **TRUE** if you want to return **NULL** columns as the literal string "**NULL**", **FALSE** if you want them returned as the empty string "". You can override this setting at runtime with **ifx_nullformat()**.

Tipos de recursos

Constantes predefinidas

Esta extensión no tiene ninguna constante definida.

Tabla de contenidos

ifx_free_blob -- Borra el objeto blob
ifx_free_char -- Elimina un objeto char
ifx_free_result -- Libera los recursos de una consulta
ifx_get_blob -- Obtiene el contenido de un objeto blob
ifx_get_char -- Obtiene el contenido de un objeto char
ifx_getsqlca -- Después de una consulta, obtiene el contenido de sqlca.sqlerrd[0..5]
ifx_htmltbl_result -- Muestra todos los registros de una consulta en una tabla HTML
ifx_nullformat -- Define el valor por defecto cuando se leen valores nulos
ifx_num_fields -- Devuelve el número de columnas en una consulta
ifx_num_rows -- Cuenta los registros ya leídos de una consulta
ifx_pconnect -- Abre una conexión permanente con Informix
ifx_prepare -- Preparar una sentencia-SQL para su ejecución
ifx_query -- Enviar una consulta Informix
ifx_textasvarchar -- Define el modo por defecto para los campos de tipo text
ifx_update_blob -- Actualiza el contenido de un objeto blob
ifx_update_char -- Actualiza el contenido de un objeto char
ifxus_close_slob -- Cierra un objeto slob
ifxus_create_slob -- Crea un objeto slob y lo abre
ifxus_free_slob -- Elimina un objeto slob
ifxus_open_slob -- Abre un objeto slob
ifxus_read_slob -- Lee un número de bytes (nbytes) de un objeto slob
ifxus_seek_slob -- Establece la posición de archivo o búsqueda actual
ifxus_tell_slob -- Devuelve la posición de archivo o búsqueda actual
ifxus_write_slob -- Escribe una cadena en un objeto slob

ifx_affected_rows

(PHP 3>= 3.0.3, PHP 4 , PHP 5)

ifx_affected_rows -- Obtiene el número de registros procesados por una consulta

Descripción

int **ifx_affected_rows** (int result_id)

result_id es un identificador válido del resultado de **ifx_query()** o **ifx_prepare()**.

Devuelve el número de filas procesadas por una consulta representada por un *result_id* (identificador de resultado).

Para inserciones, actualizaciones y borrados el número es exactamente los registros procesados (sqlerrd[2]). Para las consultas de selección es una estimación (sqlerrd[0]). No confíes en él.

Es útil llamarla después de ejecutar **ifx_prepare()** pues así podemos limitar las consultas a número razonable de registros.

Examina también: **ifx_num_rows()**

Ejemplo 1. Número de registros procesados por una consulta

```
$rid = ifx_prepare ("select * from emp where name like " . $name, $connid);
if (! $rid) {
    ... error ...
}
$rowcount = ifx_affected_rows ($rid);
if ($rowcount > 1000) {
    printf ("Too many rows in result set (%d)\n<br>", $rowcount);  // Demasiados regist
    die ("Please restrict your query<br>\n");                      // Por favor, restri
}
```

ifx_blobinfile_mode

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

ifx_blobinfile_mode -- Define el modo por defecto para los blob en todas las consultas de selección

Descripción

void **ifx_blobinfile_mode** (int mode)

Define el modo por defecto para los blob en todas las consultas de selección. El modo (mode) "0" quiere decir que guarda en memoria los blobs de tipo BYTE y modo "1" significa guardarlos en un archivo.

ifx_byteasvarchar

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

ifx_byteasvarchar -- Define el modo por defecto para los campos de tipo byte

Descripción

void **ifx_byteasvarchar** (int mode)

Define el modo por defecto para los campos de tipo byte en todas las consultas de selección. Modo (mode) "0" devolverá un identificador de blob y "1" dará el contenido en un campo de tipo varchar.

ifx_close

(PHP 3>= 3.0.3, PHP 4 , PHP 5)

ifx_close -- Cierra una conexión con Informix

Descripción

int **ifx_close** ([int link_identifier])

Devuelve: **`TRUE`** siempre.

ifx_close() cierra un enlace a una base de datos Informix que esté asociado con el identificador de enlace (link_identifier). Si el identificador de enlace no es especificado, el último enlace abierto es

asumido.

Observa que esto no es necesario habitualmente ya que las conexiones no permanentes son cerradas automáticamente al finalizar el guión (script).

ifx_close() no cerrará enlaces persistentes generados por **ifx_pconnect()**.

Examina también: **ifx_connect()**, y **ifx_pconnect()**.

Ejemplo 1. Cierre de una conexión a Informix

```
$conn_id = ifx_connect (mydb@ol_srv, "itsme", "mypassword");
... algunas consultas y código ...
ifx_close($conn_id);
```

ifx_connect

(PHP 3>= 3.0.3, PHP 4 , PHP 5)

ifx_connect -- Abrir una conexión con un servidor Informix

Descripción

int **ifx_connect** ([string base_datos [, string id_usuario [, string contrasenya]]])

Devuelve un identificador de conexión en caso de tener éxito, o **FALSE** si ocurre un fallo.

ifx_connect() establece una conexión con un servidor Informix. Todos los argumentos son opcionales, y si ellos faltan, los valores predeterminados son tomados del archivo de configuración (ifx.default_host para el host (las bibliotecas Informix usarán el valor de entorno *INFORMIXSERVER* si no se encuentra definido), ifx.default_user para el usuario, ifx.default_password para la contraseña (ninguna si no se define).

En caso de que una segunda llamada sea realizada a **ifx_connect()** con los mismos argumentos, no se establecerá un enlace nuevo, en su lugar, el identificador del enlace ya abierto será devuelto.

El enlace con el servidor será cerrado tan pronto como la ejecución del script finalice, a menos que éste sea cerrado antes mediante una llamada explícita a **ifx_close()**.

Ejemplo 1. Conexión con una base de datos Informix

```
<?php
$id_con = ifx_connect ("mi_bd@ol_srv1", "yo_mismo", "mi_contrasenya");
?>
```

Vea también **ifx_pconnect()** y **ifx_close()**.

ifx_copy_blob

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

ifx_copy_blob -- Duplica el objeto blob dado

Descripción

int **ifx_copy_blob** (int bid)

Duplica el objeto blob dado. *bid* es el identificador del objeto blob a copiar.

Devuelve **FALSE** si hubo error, en otro caso el identificador del nuevo objeto blob.

ifx_create_blob

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

ifx_create_blob -- Crea un objeto blob

Descripción

int **ifx_create_blob** (int tipo, int modo, string param)

Crea un objeto blob.

tipo: 1 = TEXT, 0 = BYTE

modo: 0 = el objeto blob mantiene el contenido en memoria, 1 = el objeto blob mantiene el contenido en un archivo.

param: si mode = 0: apuntador al contenido, si mode = 1: apuntador a la cadena de archivo.

Devuelve **FALSE** en caso de fallo, o de lo contrario un nuevo id de objeto blob.

ifx_create_char

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

ifx_create_char -- Crea un objeto char

Descripción

int **ifx_create_char** (string param)

Crea un objeto char. *param* será el contenido del char.

ifx_do

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

ifx_do -- Ejecuta una sentencia SQL preparada previamente

Descripción

int **ifx_do** (int result_id)

Devuelve **TRUE** si se realizó, **FALSE** si hubo algún error.

Ejecuta una consulta preparada anteriormente o abre un cursor para ella.

No libera *result_id* si hubo un error.

También define el número real de registros procesados para consultas que no sean de selección y se puede obtener mediante **ifx_affected_rows()**.

Examina también: **ifx_prepare()** (hay un ejemplo).

ifx_error

(PHP 3>= 3.0.3, PHP 4 , PHP 5)

ifx_error -- Devuelve el código de error de la última llamada a Informix

Descripción

string **ifx_error** (void)

Los códigos de error de Informix (SQLSTATE & SQLCODE) son representados como se especifica a continuación:

x [SQLSTATE = aa bbb SQLCODE=cccc]

donde x = un espacio : no hubo error

E : hubo error

N : no hay más datos

W : aviso

? : no definido

Si el carácter "x" es cualquier otra cosa diferente a un espacio, SQLSTATE y SQLCODE describen el error con mayor detalle.

Examina el manual de Informix para el significado de SQLSTATE y SQLCODE.

Devuelve en una cadena un caracter describiendo el resultado de una sentencia y los valores SQLSTATE y SQLCODE asociados con la última sentencia SQL ejecutada. El formato de la cadena es "(char) [SQLSTATE=(dos dígitos) (tres dígitos) SQLCODE=(un dígitos)]". El primer carácter puede ser ' ' (un espacio) (no hubo error), '*W*' (la sentencia provocó un aviso), '*E*' (la consulta produjo un error) o '*N*' (la sentencia no devolvió ningún dato).

Examina también: **ifx_errormsg()**

ifx_errormsg

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

ifx_errormsg -- Devuelve el mensaje de error de la última llamada a Informix

Descripción

string **ifx_errormsg** ([int errorcode])

Devuelve el mensaje de error asociado con el error más reciente de Informix. Si definismos el parámetro opcional "*errorcode*" (código de error), nos dará el mensaje de error correspondiente a ese código.

Examina también: **ifx_error()**

```
printf("%s\n<br>", ifx_errormsg(-201));
```

ifx_fetch_row

(PHP 3>= 3.0.3, PHP 4 , PHP 5)

ifx_fetch_row -- Obtiene registros como un array (vector) enumerado

Descripción

array **ifx_fetch_row** (int result_id [, mixed position])

Devuelve un array (vector) correspondiente a la fila leída o **FALSE** si no hay más registros.

Las columnas blob son devueltas como identificadores de blob enteros (integer) para usarlos con **ifx_get_blob()** a menos que hayas usado ifx_textasvarchar(1) o ifx_byteasvarchar(1), en cuyo caso los blobs son devueltos como cadenas de texto. Devuelve **FALSE** si hubo error.

result_id es un identificador válido del resultado de **ifx_query()** o **ifx_prepare()** (sólo para consultas de selección).

position es un parámetro opcional para una operación de lectura sobre un cursor de tipo "scroll": "NEXT" (siguiente), "PREVIOUS" (anterior), "CURRENT" (actual), "FIRST" (primero), "LAST" (último) o un número. Si se especifica un número, un registro concreto es leído. Este parámetro opcional es sólo válido para cursores de tipo scroll.

ifx_fetch_row() lee el contenido de un registro de la consulta representada por el identificador de resultado indicado. La fila (registro) es devuelta en un array. Cada columna es guarda en un array, empezando éste desde cero.

Las llamadas posteriores a **ifx_fetch_row()** devolverán el registro siguiente en el resultado de la consulta, o **FALSE** si no hay más filas.

Ejemplo 1. Leer registros

```
$rid = ifx_prepare ("select * from emp where name like " . $name,
                    $connid, IFX_SCROLL);
if (! $rid) {
    ... error ...
}
$rowcount = ifx_affected_rows($rid);
if ($rowcount > 1000) {
    printf ("Too many rows in result set (%d)\n<br>", $rowcount);  // Demasiados regist
    die ("Please restrict your query<br>\n");                      // Por favor, restri
}
if (! ifx_do ($rid)) {
   ... error ...
}
$row = ifx_fetch_row ($rid, "NEXT");
while (is_array($row)) {
    for(reset($row); $fieldname=key($row); next($row)) {
        $fieldvalue = $row[$fieldname];
        printf ("%s = %s,", $fieldname, $fieldvalue);
    }
    printf("\n<br>");
    $row = ifx_fetch_row ($rid, "NEXT");
}
ifx_free_result ($rid);
```

ifx_fieldproperties

(PHP 3>= 3.0.3, PHP 4 , PHP 5)

ifx_fieldproperties -- Indica las propiedades de los campos de una consulta SQL

Descripción

array **ifx_fieldproperties** (int result_id)

Dada una consulta representada por *result_id* devuelve un array con los nombres de campo como llaves y las propiedades como datos. **FALSE** es devuelto si hubo error.

Devuelve las propiedades SQL de cada campo como un array. Las propiedades son codificadas así: "SQLTYPE;longitud;precisión;escala;ISNULLABLE" siendo SQLTYPE el tipo de dato definido en Informix como puede ser "SQLVCHAR" etc. e ISNULLABLE (puede ser nulo) igual a "Y" sí o "N" no.

Ejemplo 1. Propiedades de los campos de una consulta SQL

```
<?php
$properties = ifx_fieldtypes ($resultid);
if (! isset($properties)) {
  ... error ...
}
for ($i = 0; $i < count($properties); $i++) {
    $fname = key ($properties);
    printf ("%s:\t type =  %s\n", $fname, $properties[$fname]);
    next ($properties);
}
?>
```

ifx_fieldtypes

(PHP 3>= 3.0.3, PHP 4 , PHP 5)

ifx_fieldtypes -- Obtiene los campos de una consulta SQL

Descripción

array **ifx_fieldtypes** (int result_id)

Dada una consulta representada por *result_id* devuelve un array con los nombres de campo como llaves y los tipos como datos. Si no tuvo éxito da **FALSE**.

Ejemplo 1. Nombres y tipos de campos de una consulta SQL

```
<?php
$types = ifx_fieldtypes ($resultid);
if (! isset ($types)) {
  ... error ...
}
for ($i = 0; $i < count($types); $i++) {
    $fname = key($types);
    printf("%s :\t type =  %s\n", $fname, $types[$fname]);
    next($types);
}
?>
```

ifx_free_blob

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

ifx_free_blob -- Borra el objeto blob

Descripción

int **ifx_free_blob** (int bid)

Elimina el objeto blob representado por el identificador *bid*. Devuelve **FALSE** si se produjo error, en otro caso **TRUE**.

ifx_free_char

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

ifx_free_char -- Elimina un objeto char

Descripción

int **ifx_free_char** (int bid)

Borra el objeto char representado por el identificador del char *bid*. Devuelve **FALSE** si se produjo un error, en otro caso **TRUE**.

ifx_free_result

(PHP 3>= 3.0.3, PHP 4 , PHP 5)

ifx_free_result -- Libera los recursos de una consulta

Descripción

int **ifx_free_result** (int result_id)

Libera los recursos representados por el identificador *result_id* de una consulta. Devuelve **`FALSE`** si hubo error.

ifx_get_blob

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

ifx_get_blob -- Obtiene el contenido de un objeto blob

Descripción

int **ifx_get_blob** (int bid)

Devuelve el contenido de un objeto blob representado por su identificador *bid*.

ifx_get_char

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

ifx_get_char -- Obtiene el contenido de un objeto char

Descripción

int **ifx_get_char** (int bid)

Devuelve el contenido de un objeto char representado por su identificador *bid*.

ifx_getsqlca

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

ifx_getsqlca -- Después de una consulta, obtiene el contenido de sqlca.sqlerrd[0..5]

Descripción

array **ifx_getsqlca** (int result_id)

result_id es un identificador válido del resultado de **ifx_query()** o **ifx_prepare()**.

Devuelve una seudo fila (array asociativo) con los valores de sqlca.sqlerrd[0] a sqlca.sqlerrd[5] de una consulta ejecutada, representada ésta con un identificador de resultado *result_id*.

Para inserciones, actualizaciones y borrados los valores devueltos son aquellos definidos por el

servidor después de que la consulta sea ejecutada. Esto da acceso al número de registros procesados y al valor de una columna de tipo serial en una consulta de inserción. Para consultas de selección, los valores son guardados cuando se prepara la sentencia. También permite conocer el número estimado de registros procesados. El uso de esta función evita el sobrecoste de ejecutar la consulta "select dbinfo('sqlca.sqlerrdx')", como obtener los valores guardados por el conector para Informix en el momento apropiado.

Ejemplo 1. Obtener los valores sqlca.sqlerrd[x]

```
/* suponiendo que la primera columna de la tabla 'sometable' es de tipo serial */
$qid = ifx_query("insert into sometable values(0, '2nd column', 'another column' ", $co
if (! $qid) {
    ... error ...
}
$sqlca = ifx_getsqlca ($qid);
$serial_value = $sqlca["sqlerrd1"];
echo "The serial value of the inserted row is : " . $serial_value<br>\n"; // El valor d
```

ifx_htmltbl_result

(PHP 3>= 3.0.3, PHP 4 , PHP 5)

ifx_htmltbl_result -- Muestra todos los registros de una consulta en una tabla HTML

Descripción

int **ifx_htmltbl_result** (int result_id [, string html_table_options])

Devuelve el número de registros leídos o **FALSE** si hubo error.

Muestra todas las filas de la consulta *result_id* dentro de una tabla html. El argumento segundo, opcional, es una cadena de parámetros del tag <table>

Ejemplo 1. Mostrar resultado como una tabla HTML

```
$rid = ifx_prepare ("select * from emp where name like " . $name,
                    $connid, IFX_SCROLL);
if (! $rid) {
   ... error ...
}
$rowcount = ifx_affected_rows ($rid);
if ($rowcount > 1000) {
    printf ("Too many rows in result set (%d)\n<br>", $rowcount);  // Demasiados regist
    die ("Please restrict your query<br>\n");                      // Por favor, restri
}
if (! ifx_do($rid) {
  ... error ...
}

ifx_htmltbl_result ($rid, "border=\"2\"");

ifx_free_result($rid);
```

ifx_nullformat

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

ifx_nullformat -- Define el valor por defecto cuando se leen valores nulos

Descripción

void **ifx_nullformat** (int mode)

Define el valor por defecto cuando se leen valores nulos. Modo (mode) "0" devuelve "", y modo "1" devuelve "**NULL**".

ifx_num_fields

(PHP 3>= 3.0.3, PHP 4 , PHP 5)

ifx_num_fields -- Devuelve el número de columnas en una consulta

Descripción

int **ifx_num_fields** (int result_id)

Dada una consulta representada por *result_id* devuelve el número de columnas o **FALSE** si se produjo un error.

Después de preparar o ejecutar una consulta, una llamada a esta función te da el número de columnas en la consulta.

ifx_num_rows

(PHP 3>= 3.0.3, PHP 4 , PHP 5)

ifx_num_rows -- Cuenta los registros ya leídos de una consulta

Descripción

int **ifx_num_rows** (int result_id)

Da el número de registros ya leídos de una consulta representada por un *result_id* después de llamar a **ifx_query()** o **ifx_do()**.

ifx_pconnect

(PHP 3>= 3.0.3, PHP 4 , PHP 5)

ifx_pconnect -- Abre una conexión permanente con Informix

Descripción

int **ifx_pconnect** ([string database [, string userid [, string password]]])

Devuelve un identificador positivo de enlace persistente si hubo conexión, o **FALSE** si se produjo un error.

ifx_pconnect() actúa muy parecido a **ifx_connect()** con dos principales diferencias.

Esta función se comporta exactamente igual que **ifx_connect()** cuando PHP no es ejecutado como un módulo de Apache. La primera diferencia es cuando se conecta, la función intentará encontrar un enlace (persistente) que exista con el mismo servidor, usuario y contraseña. Si es hallado, el identificador del enlace será devuelto en vez de abrir una nueva conexión.

Segundo, la conexión al servidor no se cerrará cuando la ejecución del guión (script) finalice. En vez de esto, la conexión permanecerá abierta para usos futuros (**ifx_close()** no cerrará el enlace creado por **ifx_pconnect()**).

Este tipo de enlace es, por tanto, llamado 'persistente'

Examina también: **ifx_connect()**.

ifx_prepare

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

ifx_prepare -- Preparar una sentencia-SQL para su ejecución

Descripción

int **ifx_prepare** (string consulta, int id_con [, int tipo_cursor, mixed matriz_id_blob])

Devuelve un entero *id_resultado* para su uso por **ifx_do()**. Define *filas_afectadas* para su recuperación a través de la función **ifx_affected_rows()**.

Prepara *consulta* sobre la conexión *id_con*. Para consultas "tipo select", se declara y abre un cursor. El parámetro opcional *tipo_cursor* le permite hacer de éste un cursor "scroll" o "hold". Es una máscara de bits y puede ser IFX_SCROLL, IFX_HOLD, o ambos, unidos por el operador lógico OR.

Para cualquiera de los tipos de consulta, el número estimado de filas afectadas es almacenado para su recuperación por **ifx_affected_rows()**.

Si tiene columnas BLOB (BYTE o TEXT) en la consulta, puede agregar un parámetro *matriz_id_blob* que contenga los "ids blob" correspondientes, y debería reemplazar esas columnas con un "?" en el texto de la consulta.

Si los contenidos de la columna TEXT (o BYTE) lo permiten, puede usar también "ifx_textasvarchar(1)" y "ifx_byteasvarchar(1)". Esto le permite tratar columnas TEXT (o BYTE) como si fueran columnas VARCHAR ordinarias (pero largas) para las consultas select, y no debe preocuparse por id's blob.

Con ifx_textasvarchar(0) o ifx_byteasvarchar(0) (la situación predeterminada), las consultas select devolverán columnas BLOB como id's blob (valor entero). Puede obtener el valor del blob como una cadena o un archivo con las funciones blob (vea más adelante).

Vea también: **ifx_do()**.

ifx_query

(PHP 3>= 3.0.3, PHP 4 , PHP 5)

ifx_query -- Enviar una consulta Informix

Descripción

int **ifx_query** (string consulta, int id_enlace [, int tipo_cursor [, mixed matriz_id_blob]])

Devuelve un identificador de resultado positivo si tuvo éxito, o **FALSE** si ocurrió un error.

Un recurso "id_resultado" es usado por otras funciones para obtener los resultados de la consulta. Define "filas_afectadas" para su consulta a través de la función **ifx_affected_rows()**.

ifx_query() envía una consulta a la base de datos activa actualmente en el servidor, la cual está asociada con el identificador de enlace dado.

Ejecuta *consulta* sobre la conexión *id_enlace*. En el caso de consultas "tipo-select", un cursor es declarado y abierto. El parámetro opcional *tipo_cursor* le permite hacer que éste sea un cursor "scroll" o "hold". Es una máscara de bits y puede ser IFX_SCROLL, IFX_HOLD, o ambos unidos mediante la operación lógica OR. Las consultas que no son de selección son de "ejecución inmediata". IFX_SCROLL e IFX_HOLD son constantes simbólicas, y como tales no deben encontrarse entre comillas. Si omite este parámetro, el cursor será un cursor secuencial normal.

Para cualquier tipo de consulta, el número (estimado o real) de filas afectadas es almacenado para su consulta mediante **ifx_affected_rows()**.

Si tiene columnas BLOB (BYTE o TEXT) en una consulta de actualización, puede añadir un parámetro *matriz_id_blob* que contenga los "ids de blob" correspondientes, y debería reemplazar esas columnas con un "?" en el texto de la consulta.

Si el contenido de la columna TEXT (o BYTE) lo permite, puede usar también "ifx_textasvarchar (1)" y "ifx_byteasvarchar(1)". Esto le permite tratar columnas TEXT (o BYTE) como si fueran columnas normales (pero largas) de tipo VARCHAR para consultas select, y no necesita complicarse con id's de blob.

Con ifx_textasvarchar(0) o ifx_byteasvarchar(0) (la situación predeterminada), las consultas select devolverán columnas BLOB como id's blob (valores enteros). Puede obtener el valor del blob como una cadena o archivo con las funciones blob (vea más adelante).

Ejemplo 1. Mostrar todas las filas de la tabla "ordenes" como una tabla HTML

```
<?php
ifx_textasvarchar(1);      // usar "modo de texto" para blobs
$id_res = ifx_query("select * from ordenes", $id_con);
if (! $id_res) {
    printf("No se pueden seleccionar &oacute;rdenes: %s\n<br />%s<br />\n", ifx_error()
    ifx_errormsg();
    die;
}
ifx_htmltbl_result($id_res, "border=\"1\"");
ifx_free_result($id_res);
?>
```

Ejemplo 2. Insertar algunos valores en la tabla "catalogo"

```
<?php

// crear id's blob para una columna byte y text
$textid = ifx_create_blob(0, 0, "Text column in memory");
$byteid = ifx_create_blob(1, 0, "Byte column in memory");

// almacenar id's blob en una matriz id_blob
$matriz_id_blob[] = $textid;
$matriz_id_blob[] = $byteid;

// ejecutar consulta
$consulta = "insert into catalogo (num_bodega, cod_manu, " .
         "descr_cat,imagen_cat) values(1,'HRO',?,?)";
$id_res = ifx_query($consulta, $id_con, $matriz_id_blob);
if (! $id_res) {
    /* ... error ... */
}

// liberar el id de resultado
ifx_free_result($id_res);
?>
```

Vea también **ifx_connect()**.

ifx_textasvarchar

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

ifx_textasvarchar -- Define el modo por defecto para los campos de tipo text

Descripción

void **ifx_textasvarchar** (int mode)

Define el modo por defecto para los campos de tipo text en todas las consultas de selección. Modo (mode) "0" devolverá un identificador de blob y "1" dará el contenido en un campo de tipo varchar.

ifx_update_blob

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

ifx_update_blob -- Actualiza el contenido de un objeto blob

Descripción

ifx_update_blob (int bid, string content)

Actualiza el contenido de un objeto blob representado por su identificador *bid*. *content* es una cadena con el nuevo contenido. Devuelve **FALSE** si hubo error, en otro caso **TRUE**.

ifx_update_char

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

ifx_update_char -- Actualiza el contenido de un objeto char

Descripción

int **ifx_update_char** (int bid, string content)

Actualiza el contenido de un objeto char representado por su identificador *bid*. *content* es una cadena con la información nueva. Devuelve **FALSE** si se produjo un error, en otro caso **TRUE**.

ifxus_close_slob

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

ifxus_close_slob -- Cierra un objeto slob

Descripción

int **ifxus_close_slob** (int bid)

Cierra un objeto slob representado por su identificador de slob *bid*. Devuelve **FALSE** si hubo error, **TRUE** en otro caso.

ifxus_create_slob

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

ifxus_create_slob -- Crea un objeto slob y lo abre

Descripción

int **ifxus_create_slob** (int mode)

Crea un objeto slob y lo abre. Modos: 1 = LO_RDONLY, 2 = LO_WRONLY, 4 = LO_APPEND, 8 = LO_RDWR, 16 = LO_BUFFER, 32 = LO_NOBUFFER o una combinación de ellos. También puedes usar nombres de constantes IFX_LO_RDONLY, IFX_LO_WRONLY, etc. Devuelve **FALSE** si hubo error, en otro caso el identificador del nuevo objeto slob.

ifxus_free_slob

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

ifxus_free_slob -- Elimina un objeto slob

Descripción

int **ifxus_free_slob** (int bid)

Borra un objeto slob. *bid* es el identificador del objeto slob. Devuelve **FALSE** si hubo error, **TRUE** en otro caso.

ifxus_open_slob

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

ifxus_open_slob -- Abre un objeto slob

Descripción

int **ifxus_open_slob** (long bid, int mode)

Abre un objeto slob. *bid* será un identificador de slob que válido. Modos: 1 = LO_RDONLY, 2 = LO_WRONLY, 4 = LO_APPEND, 8 = LO_RDWR, 16 = LO_BUFFER, 32 = LO_NOBUFFER o una combinación de ellos. Devuelve **FALSE** si hubo error, en otro caso el identificador del nuevo objeto slob.

ifxus_read_slob

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

ifxus_read_slob -- Lee un número de bytes (nbytes) de un objeto slob

Descripción

int **ifxus_read_slob** (long bid, long nbytes)

Lee un número de bytes (nbytes) de un objeto slob. *bid* es un identificador de slob válido y *nbytes* es el número de bytes a leer. Devuelve **FALSE** si hubo error, sino la cadena.

ifxus_seek_slob

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

ifxus_seek_slob -- Establece la posición de archivo o búsqueda actual

Descripción

int **ifxus_seek_slob** (int bid, int modo, int desplazamiento)

Establece la posición de archivo o búsqueda actual de un objeto slob abierto. *bid* debe ser un ID de un slob existente. Modos: 0 = LO_SEEK_SET, 1 = LO_SEEK_CUR, 2 = LO_SEEK_END y *desplazamiento* es un desplazamiento en bytes. Devuelve **FALSE** en caso de error, o la posición de búsqueda en cualquier otro caso.

ifxus_tell_slob

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

ifxus_tell_slob -- Devuelve la posición de archivo o búsqueda actual

Descripción

int **ifxus_tell_slob** (int bid)

Devuelve la posición de archivo o búsqueda actual de un objeto slob abierto, *bid* debe ser un ID de un slob existente. Devuelve **FALSE** en caso de fallo, o en cualquier otro caso la posición de búsqueda.

ifxus_write_slob

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

ifxus_write_slob -- Escribe una cadena en un objeto slob

Descripción

int **ifxus_write_slob** (long bid, string content)

Escribe una cadena en un objeto slob. *bid* es un identificador de slob válido y *content* el contenido a escribir. Devuelve **FALSE** si hubo error, sino el número de bytes escritos.

LI. IIS Administration Functions

Introducción

Esta extension PECL no esta ligada a PHP. This extension is available for Win32 only. It provides functions to administrate Microsoft Internet Information Server (IIS). The extension includes function to create web sites and virtual directories as well as configuring security and script mapping. These functions have been added in PHP 4.

In order to use these functions you must enable the `php_iisfunc.dll` DLL inside of `php.ini`. Podeis descargar esta DLL de las extensiones PECL desde la pagina PHP Downloads o desde http://snaps.php.net/.

Requirimientos

No se necesitan bibliotecas externas para construir esta extensión

---

Instalación

No se necesita ninguna instalación para usar estas funciones, son parte del núcleo de PHP.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Esta extensión no tiene ninguna constante definida.

Tabla de contenidos

iis_add_server

(no version information, might be only in CVS)

iis_add_server -- Creates a new virtual web server

Description

int **iis_add_server** (string path, string comment, string server_ip, int port, string host_name, int rights, int start_server)

iis_get_dir_security

(no version information, might be only in CVS)

iis_get_dir_security -- Gets Directory Security

Description

int **iis_get_dir_security** (int server_instance, string virtual_path)

iis_get_script_map

(no version information, might be only in CVS)

iis_get_script_map -- Gets script mapping on a virtual directory for a specific extension

Description

int **iis_get_script_map** (int server_instance, string virtual_path, string script_extension)

iis_get_server_by_comment

(no version information, might be only in CVS)

iis_get_server_by_comment -- Return the instance number associated with the Comment

Description

int **iis_get_server_by_comment** (string comment)

iis_get_server_by_path

(no version information, might be only in CVS)

iis_get_server_by_path -- Return the instance number associated with the Path

Description

int **iis_get_server_by_path** (string path)

Each virtual server in IIS is associated with an instance number. **iis_get_server_by_path()** Finds the instance number from the actual path to the root directory.

iis_get_server_rights

(no version information, might be only in CVS)

iis_get_server_rights -- Gets server rights

Description

int **iis_get_server_rights** (int server_instance, string virtual_path)

iis_get_service_state

(no version information, might be only in CVS)

iis_get_service_state -- Starts the service defined by ServiceId

Description

int **iis_get_service_state** (string service_id)

iis_remove_server

(no version information, might be only in CVS)

iis_remove_server -- Removes the virtual web server indicated by ServerInstance

Description

int **iis_remove_server** (int server_instance)

iis_set_app_settings

(no version information, might be only in CVS)

iis_set_app_settings -- Creates application scope for a virtual directory

Description

int **iis_set_app_settings** (int server_instance, string virtual_path, string application_scope)

iis_set_dir_security

(no version information, might be only in CVS)

iis_set_dir_security -- Sets Directory Security

Description

int **iis_set_dir_security** (int server_instance, string virtual_path, int directory_flags)

iis_set_script_map

(no version information, might be only in CVS)

iis_set_script_map -- Sets script mapping on a virtual directory

Description

int **iis_set_script_map** (int server_instance, string virtual_path, string script_extension, string engine_path, int allow_scripting)

iis_set_server_rights

(no version information, might be only in CVS)

iis_set_server_rights -- Sets server rights

Description

int **iis_set_server_rights** (int server_instance, string virtual_path, int directory_flags)

iis_start_server

(no version information, might be only in CVS)

iis_start_server -- Starts the virtual web server

Description

int **iis_start_server** (int server_instance)

iis_start_service

(no version information, might be only in CVS)

iis_start_service -- Starts the service defined by ServiceId

Description

int **iis_start_service** (string service_id)

iis_stop_server

(no version information, might be only in CVS)

iis_stop_server -- Stops the virtual web server

Description

int **iis_stop_server** (int server_instance)

iis_stop_service

(no version information, might be only in CVS)

iis_stop_service -- Stops the service defined by ServiceId

Description

int **iis_stop_service** (string service_id)

LII. Funciones para imágenes

Puede usar las funciones para imágenes en PHP para obtener el tamaño de imágenes JPEG, GIF, PNG, SWF, TIFF y JPEG2000, y si tiene instalada la biblioteca GD (disponible en http://www.boutell.com/gd/) también podrá crear y manipular imágenes.

Si tiene PHP compilado con *--enable-exif* puede trabajar con la información guardada en las cabeceras de las imágenes JPEG y TIFF. Estas funciones no requieren la biblioteca GD.

Los formatos de imágenes que puede manipular dependen de la versión de GD que instale y de cualquier otra biblioteca que GD pueda necesitar para acceder a esos formatos de imagen. Las versiones de GD anteriores a la GD-1.6 soportan imágenes en formato gif y no soportan png, mientras que las versiones superiores a la GD-1.6 soportan el formato png y no el gif.

Antes de poder leer y escribir imágenes en formato jpeg, deberá obtener e instalar jpeg-6b (disponible en ftp://ftp.uu.net/graphics/jpeg/), y después recompilar GD para poder hacer uso de jpeg-6b. También tendrá que compilar PHP con la opción *--with-jpeg-dir=/ruta/a/jpeg-6b*.

Para añadir el soporte de fuentes Type 1, puede instalar t1lib (disponible en ftp://sunsite.unc.edu/pub/Linux/libs/graphics/), y entonces añadir la opción *--with-t1lib[=dir]* al recompilar.

Tabla de contenidos

ImageColorExact -- Devuelve el índice del color especificado
imagecolorexactalpha -- Get the index of the specified color + alpha
imagecolormatch -- Makes the colors of the palette version of an image more closely match the true color version
ImageColorResolve -- Devuelve el índice del color especificado o su posible alternativa más cercana
imagecolorresolvealpha -- Get the index of the specified color + alpha or its closest possible alternative
ImageColorSet -- Establece el color para el índice de la paleta especificado
ImageColorsForIndex -- Obtiene los colores de un índice
ImageColorsTotal -- Encuentra el número de colores de la paleta de una imagen
ImageColorTransparent -- Define un color como transparente
imagecopy -- Copy part of an image
imagecopymerge -- Copy and merge part of an image
imagecopymergegray -- Copy and merge part of an image with gray scale
imagecopyresampled -- Copia y reescala parte de una imagen con remuestreo
ImageCopyResized -- Copia y redimensiona parte de una imagen
ImageCreate -- Crea una nueva imagen
imagecreatefromgd2 -- Create a new image from GD2 file or URL
imagecreatefromgd2part -- Create a new image from a given part of GD2 file or URL
imagecreatefromgd -- Create a new image from GD file or URL
imagecreatefromgif -- Crear una nueva imagen a partir de un archivo o URL
imagecreatefromjpeg -- Crea una imagen nueva desde un archivo o URL
imagecreatefrompng -- Crea una imagen nueva desde un archivo o URL
imagecreatefromstring -- Create a new image from the image stream in the string
imagecreatefromwbmp -- Create a new image from file or URL
imagecreatefromxbm -- Create a new image from file or URL
imagecreatefromxpm -- Create a new image from file or URL
imagecreatetruecolor -- Crea una imagen nueva en color real (true color)
ImageDashedLine -- Dibuja una línea discontinua
ImageDestroy -- Destruye una imagen
imageellipse -- Draw an ellipse
ImageFill -- Relleno
imagefilledarc -- Draw a partial ellipse and fill it
imagefilledellipse -- Draw a filled ellipse
ImageFilledPolygon -- Dibuja un polígono relleno
ImageFilledRectangle -- dibuja un rectángulo relleno
ImageFillToBorder -- Relleno de un color especifico
imagefilter -- Applies a filter to an image
ImageFontHeight -- Devuelve la altura de una fuente
ImageFontWidth -- Devuelve la anchura de una fuente
imageftbbox -- Give the bounding box of a text using fonts via freetype2
imagefttext -- Write text to the image using fonts using FreeType 2
imagegammacorrect -- Apply a gamma correction to a GD image
imagegd2 -- Output GD2 image to browser or file
imagegd -- Output GD image to browser or file
imagegif -- Producir la salida de una imagen al navegador o a un archivo
ImageInterlace -- Activa o desactiva el entrelazado
imageistruecolor -- Finds whether an image is a truecolor image
imagejpeg -- Output image to browser or file
imagelayereffect -- Set the alpha blending flag to use the bundled libgd layering effects
ImageLine -- Dibuja una línea
imageloadfont -- Cargar una fuente nueva
imagepalettecopy -- Copy the palette from one image to another
imagepng -- Output a PNG image to either the browser or a file

ImagePolygon -- Dibuja un polígono
ImagePSBBox -- Devuelve el borde que rodea un rectángulo de texto usando fuentes PostScript Type1
ImagePSCopyFont -- hace una copia de una fuente ya cargada para futuras modificaciones
ImagePSEncodeFont -- Cambia el vector de codificación de caracteres de una fuente
imagepsextendfont -- Extend or condense a font
ImagePSFreeFont -- Libera la memoria usada por un fuente PostScript Type 1
imagepsloadfont -- Cargar una fuente PostScript Tipo 1 desde un archivo
imagepsslantfont -- Slant a font
ImagePSText -- Para dibujar una cadena de texto sobre una imagen usando fuentes PostScript Type1
ImageRectangle -- Dibuja un rectángulo
imagerotate -- Rotate an image with a given angle
imagesavealpha -- Set the flag to save full alpha channel information (as opposed to single-color transparency) when saving PNG images
imagesetbrush -- Set the brush image for line drawing
ImageSetPixel -- Dibuja un pixel
imagesetstyle -- Set the style for line drawing
imagesetthickness -- Set the thickness for line drawing
imagesettile -- Set the tile image for filling
ImageString -- Dibuja una cadena de texto horizintalmente
ImageStringUp -- Dibuja una cadena de texto verticalmente
ImageSX -- Obtiene la anchura de la imagen
ImageSY -- Obtiene la altura de la imagen
imagetruecolortopalette -- Convert a true color image to a palette image
imagettfbbox -- Entrega la caja circundante de un texto usando fuentes TrueType
imagettftext -- Escribir un texto sobre la imagen usando fuentes TrueType
imagetypes -- Return the image types supported by this PHP build
imagewbmp -- Output image to browser or file
imagexbm -- Output XBM image to browser or file
iptcembed -- Embed binary IPTC data into a JPEG image
iptcparse -- Parse a binary IPTC http://www.iptc.org/ block into single tags.
jpeg2wbmp -- Convert JPEG image file to WBMP image file
png2wbmp -- Convert PNG image file to WBMP image file

gd_info

(PHP 4 >= 4.3.0, PHP 5)

gd_info -- Retrieve information about the currently installed GD library

Description

array **gd_info** (void)

Returns an associative array describing the version and capabilities of the installed GD library.

Tabla 1. Elements of array returned by gd_info()

Attribute	Meaning
GD Version	**string** value describing the installed *libgd* version.

Attribute	Meaning
Freetype Support	**boolean** value. **TRUE** if Freetype Support is installed.
Freetype Linkage	**string** value describing the way in which Freetype was linked. Expected values are: 'with freetype', 'with TTF library', and 'with unknown library'. This element will only be defined if *Freetype Support* evaluated to **TRUE**.
T1Lib Support	**boolean** value. **TRUE** if *T1Lib* support is included.
GIF Read Support	**boolean** value. **TRUE** if support for *reading GIF* images is included.
GIF Create Support	**boolean** value. **TRUE** if support for *creating GIF* images is included.
JPG Support	**boolean** value. **TRUE** if *JPG* support is included.
PNG Support	**boolean** value. **TRUE** if *PNG* support is included.
WBMP Support	**boolean** value. **TRUE** if *WBMP* support is included.
XBM Support	**boolean** value. **TRUE** if *XBM* support is included.

Ejemplo 1. Using gd_info()

```
<?php
var_dump(gd_info());
?>
```

The typical output is :

```
array(9) {
  ["GD Version"]=>
  string(24) "bundled (2.0 compatible)"
  ["FreeType Support"]=>
  bool(false)
  ["T1Lib Support"]=>
  bool(false)
  ["GIF Read Support"]=>
  bool(true)
  ["GIF Create Support"]=>
  bool(false)
  ["JPG Support"]=>
  bool(false)
  ["PNG Support"]=>
  bool(true)
  ["WBMP Support"]=>
  bool(true)
  ["XBM Support"]=>
  bool(false)
}
```

See also **imagepng()**, **imagejpeg()**, **imagegif()**, **imagewbmp()**, and **imagetypes()**.

getimagesize

(PHP 3, PHP 4 , PHP 5)

getimagesize -- Obtener el tamaño de una imagen

Descripción

array **getimagesize** (string nombre_archivo [, array &info_imagen])

La función **getimagesize()** determinará el tamaño de cualquier archivo de imagen GIF, JPG, PNG, SWF, SWC, PSD, TIFF, BMP, IFF, JP2, JPX, JB2, JPC, XBM, o WBMP y devuelve las dimensiones, junto con el tipo de archivo y una cadena de texto de altura/ancho a ser usada al interior de una etiqueta HTML *IMG* normal.

Si no es posible acceder a la imagen *nombre_archivo*, o si no es una imagen válida, **getimagesize()** devolverá **`FALSE`** y generará un error de nivel *E_WARNING*.

> **Nota:** El soporte para JPC, JP2, JPX, JB2, XBM, y WBMP apareció en PHP 4.3.2. El soporte para SWC existe a partir de PHP 4.3.0 y el soporte TIFF fue agregado en PHP 4.2.0
>
> **Nota:** El soporte de JPEG 2000 fue agregado en PHP 4.3.2. Note que JPC y JP2 son capaces de tener componentes con diferentes profundidades de bit. En este caso, el valor para "bits" es la profundidad de bit más alta encontrada. Asimismo, los archivos JP2 pueden contener múltiples secuencias de código JPEG 2000. En este caso, **getimagesize ()** devuelve los valores para la primera secuencia de código que encuentra en la raíz del archivo.
>
> **Nota:** La función **getimagesize()** no requiere de la biblioteca de imágenes GD.

Devuelve una matriz con 4 elementos. El índice 0 contiene el ancho de la imagen en píxeles. El índice 1 contiene la altura. El índice 2 es una bandera que indica el tipo de imagen: 1 = GIF, 2 = JPG, 3 = PNG, 4 = SWF, 5 = PSD, 6 = BMP, 7 = TIFF(orden de bytes intel), 8 = TIFF(orden de bytes motorola), 9 = JPC, 10 = JP2, 11 = JPX, 12 = JB2, 13 = SWC, 14 = IFF, 15 = WBMP, 16 = XBM. Estos valores corresponden a las constantes IMAGETYPE que fueron agregadas en PHP 4.3.0. El índice 3 es una cadena de texto con el valor correcto height="yyy" width="xxx" que puede ser usado directamente en una etiqueta IMG.

Ejemplo 1. getimagesize (archivo)

```
<?php
list($ancho, $altura, $tipo, $atr) = getimagesize("img/bandera.jpg");
echo "<img src=\"img/bandera.jpg\" $attr alt=\"ejemplo de getimagesize()\" />";
?>
```

El soporte para URLs fue agregado en PHP 4.0.5

Ejemplo 2. getimagesize (URL)

```
<?php
$tam = getimagesize("http://www.example.com/gifs/logo.gif");

// si el nombre de archivo contiene un espacio, codificarlo apropiadamente
$tam = getimagesize("http://www.example.com/gifs/lo%20go.gif");

?>
```

Con las imágenes JPG se devuelven dos índices adicionales: *channels* y *bits*. *channels* será 3 para fotos RGB y 4 para fotos CMYK. *bits* es el número de bits por cada color.

A partir de PHP 4.3.0, *bits* y *channels* se encuentran presentes para otros tipos de imágenes también. Sin embargo, la presencia de éstos valores puede ser un poco confusa. Como un ejemplo, GIF siempre usa 3 canales por píxel, pero el número de bits por píxel no puede ser calculado para un GIF animado con una tabla de colores global.

Puede que algunos formatos no contengan una imagen o que contengan múltiples imágenes. En

estos casos, puede que **getimagesize()** no sea capaz de determinar el tamaño de la imagen apropiadamente. **getimagesize()** devolverá cero como valor de ancho y altura en tales casos.

A partir de PHP 4.3.0, **getimagesize()** devuelve también un parámetro adicional, *mime*, que corresponde con el tipo MIME de la imagen. Esta información puede ser usada para despachar imágenes con las cabeceras HTTP Content-type correctas:

Ejemplo 3. getimagesize() y los tipos MIME

```
<?php
$tam = getimagesize($nombre_archivo);
$da=fopen($nombre_archivo, "rb");
if ($tam && $da) {
  header("Content-type: {$tam['mime']}");
  fpassthru($da);
  exit;
} else {
  // error
}
?>
```

El parámetro opcional *info_imagen* le permite extraer algunos datos extendidos desde el archivo de imagen. En la actualidad, esto devolverá los diferentes marcadores APP de JPG como una matriz asociativa. Algunos programas usan estos marcadores para embeber información de texto en imágenes. Un uso muy común es el embeber información IPTC http://www.iptc.org/ en el marcador APP13. Puede usar la función **iptcparse()** para convertir el marcador APP13 binario hacia algo legible.

Ejemplo 4. getimagesize() devolviendo IPTC

```
<?php
$tam = getimagesize("imagen_prueba.jpg", $info);
if (isset($info["APP13"])) {
    $iptc = iptcparse($info["APP13"]);
    var_dump($iptc);
}
?>
```

Vea también **image_type_to_mime_type()**, **exif_imagetype()**, **exif_read_data()** y **exif_thumbnail()**.

image_type_to_extension

(no version information, might be only in CVS)

image_type_to_extension -- Get file extension for image type

Description

string **image_type_to_extension** (int imagetype [, bool include_dot])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

image_type_to_mime_type

(PHP 4 >= 4.3.0, PHP 5)

image_type_to_mime_type -- Get Mime-Type for image-type returned by getimagesize, exif_read_data, exif_thumbnail, exif_imagetype

Description

string **image_type_to_mime_type** (int imagetype)

The **image_type_to_mime_type()** function will determine the Mime-Type for an IMAGETYPE constant.

Ejemplo 1. image_type_to_mime_type (file)

```
<?php
header("Content-type: " . image_type_to_mime_type(IMAGETYPE_PNG));
?>
```

The returned values are as follows

Tabla 1. Returned values Constants

imagetype	Returned value
IMAGETYPE_GIF	*image/gif*
IMAGETYPE_JPEG	*image/jpeg*
IMAGETYPE_PNG	*image/png*
IMAGETYPE_SWF	*application/x-shockwave-flash*
IMAGETYPE_PSD	*image/psd*
IMAGETYPE_BMP	*image/bmp*
IMAGETYPE_TIFF_II (intel byte order)	*image/tiff*
IMAGETYPE_TIFF_MM (motorola byte order)	*image/tiff*
IMAGETYPE_JPC	*application/octet-stream*
IMAGETYPE_JP2	*image/jp2*
IMAGETYPE_JPX	*application/octet-stream*
IMAGETYPE_JB2	*application/octet-stream*
IMAGETYPE_SWC	*application/x-shockwave-flash*
IMAGETYPE_IFF	*image/iff*
IMAGETYPE_WBMP	*image/vnd.wap.wbmp*
IMAGETYPE_XBM	*image/xbm*

Nota: This function does not require the GD image library.

See also **getimagesize()**, **exif_imagetype()**, **exif_read_data()** and **exif_thumbnail()**.

image2wbmp

(PHP 4 >= 4.0.5, PHP 5)

image2wbmp -- Output image to browser or file

Description

int **image2wbmp** (resource image [, string filename [, int threshold]])

image2wbmp() creates the WBMP file in filename from the image *image*. The *image* argument is the return from **imagecreate()**.

The filename argument is optional, and if left off, the raw image stream will be output directly. By sending an image/vnd.wap.wbmp content-type using **header()**, you can create a PHP script that outputs WBMP images directly.

Ejemplo 1. image2wbmp() example

```
<?php

$file = 'php.png';
$image = imagecreatefrompng($file);

header('Content-type: ' . image_type_to_mime_type(IMAGETYPE_WBMP));
image2wbmp($image); // output the stream directly

?>
```

Nota: WBMP support is only available if PHP was compiled against GD-1.8 or later.

See also **imagewbmp()**.

imagealphablending

(PHP 4 >= 4.0.6, PHP 5)

imagealphablending -- Set the blending mode for an image

Description

bool **imagealphablending** (resource image, bool blendmode)

imagealphablending() allows for two different modes of drawing on truecolor images. In blending mode, the alpha channel component of the color supplied to all drawing function, such as **imagesetpixel()** determines how much of the underlying color should be allowed to shine through. As a result, gd automatically blends the existing color at that point with the drawing color, and stores the result in the image. The resulting pixel is opaque. In non-blending mode, the drawing color is copied literally with its alpha channel information, replacing the destination pixel. Blending mode is not available when drawing on palette images. If *blendmode* is **TRUE**, then blending mode is enabled, otherwise disabled. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Nota: Esta funcion requiere GD 2.0.1 o posterior.

imageantialias

(PHP 4 >= 4.3.2, PHP 5)

imageantialias -- Should antialias functions be used or not

Description

bool **imageantialias** (resource im, bool on)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

Nota: Esta funcion esta disponible solamente si PHP se ha compilado con la version de las bibliotecas GD distribuidas con PHP.

See also **imagecreatetruecolor()**.

ImageArc

(PHP 3, PHP 4 , PHP 5)

ImageArc -- Dibuja una elipse parcial

Descripción

int **imagearc** (int im, int cx, int cy, int w, int h, int s, int e, int col)

ImageArc dibuja una elipse parcial centrada en cx, cy (la esquina superior izquierda es 0,0) en la imagen que representa im. w y h especifican la anchura y altura respectivamente mientras que los puntos de inicio y final vienen indicados por los parámetros s y e en grados.

ImageChar

(PHP 3, PHP 4 , PHP 5)

ImageChar -- Dibuja un carácter horizontalmente

Descripción

int **imagechar** (int im, int font, int x, int y, string c, int col)

ImageChar dibuja el primer carácter de *c* en la imagen identificada como *id* con su esquina superior izquierda en *x*, *y* (arriba izquierda es 0,0) con el color *col*. Si la fuente es 1, 2, 3, 4 o 5 se usa una fuente predefinida (números mayores corresponden con tamaños mayores).

Vea también **imageloadfont()**.

ImageCharUp

(PHP 3, PHP 4 , PHP 5)

ImageCharUp -- Dibuja un carácter vertical

Descripción

int **imagecharup** (int im, int font, int x, int y, string c, int col)

ImageCharUp dibuja el caracter c verticalmente en la imagen identificado como im en las coordenadas x, y (arriba izquierda es 0,0) con el color col. Si la fuente es 1, 2, 3, 4 o 5 se usa una fuente predefinida.

Vea también **imageloadfont()**.

ImageColorAllocate

(PHP 3, PHP 4 , PHP 5)

ImageColorAllocate -- Reserva un color para una imagen

Descripción

int **imagecolorallocate** (int im, int red, int green, int blue)

ImageColorAllocate devuelve un identificador del color representado por la mezcla de los componentes RGB dados. El parámetro im es el resultado de la función **imagecreate()**. ImageColorAlocate tiene que ser invocada para crear cada color que va a ser usado por la imagen que representa im.

```
$white = ImageColorAllocate($im, 255,255,255);
$black = ImageColorAllocate($im, 0,0,0);
```

imagecolorallocatealpha

(PHP 4 >= 4.3.2, PHP 5)

imagecolorallocatealpha -- Allocate a color for an image

Description

int **imagecolorallocatealpha** (resource image, int red, int green, int blue, int alpha)

imagecolorallocatealpha() behaves identically to **imagecolorallocate()** with the addition of the transparency parameter *alpha* which may have a value between *0* and *127*. *0* indicates completely opaque while *127* indicates completely transparent.

Returns **FALSE** if the allocation failed.

Ejemplo 1. Example of using imagecolorallocatealpha()

```php
<?php
$size = 300;
$image=imagecreatetruecolor($size, $size);

// something to get a white background with black border
$back = imagecolorallocate($image, 255, 255, 255);
$border = imagecolorallocate($image, 0, 0, 0);
imagefilledrectangle($image, 0, 0, $size - 1, $size - 1, $back);
imagerectangle($image, 0, 0, $size - 1, $size - 1, $border);

$yellow_x = 100;
$yellow_y = 75;
$red_x    = 120;
$red_y    = 165;
$blue_x   = 187;
$blue_y   = 125;
$radius   = 150;

// allocate colors with alpha values
$yellow = imagecolorallocatealpha($image, 255, 255, 0, 75);
$red    = imagecolorallocatealpha($image, 255, 0, 0, 75);
$blue   = imagecolorallocatealpha($image, 0, 0, 255, 75);

// drawing 3 overlapped circle
imagefilledellipse($image, $yellow_x, $yellow_y, $radius, $radius, $yellow);
imagefilledellipse($image, $red_x, $red_y, $radius, $radius, $red);
imagefilledellipse($image, $blue_x, $blue_y, $radius, $radius, $blue);

// don't forget to output a correct header!
header('Content-type: image/png');

// and finally, output the result
imagepng($image);
imagedestroy($image);
?>
```

Nota: Esta funcion requiere GD 2.0.1 o posterior.

See also **imagecolorallocate()** and **imagecolordeallocate()**.

ImageColorAt

(PHP 3, PHP 4 , PHP 5)

ImageColorAt -- Obtiende el índice del color de un pixel

Descripción

int **imagecolorat** (int im, int x, int y)

Devuelve el índice del color del pixel especificado en la posición de la imagen.

Vea también **imagecolorset()** y **imagecolorsforindex()**.

ImageColorClosest

(PHP 3, PHP 4 , PHP 5)

ImageColorClosest -- Obtiene el índice del color más cercano al color especificado

Descripción

int **imagecolorclosest** (int im, int red, int green, int blue)

Devuelve el índice del color de la paleta de la imagen que sea más "cercano" al valor RGB especificado.

La "distancia" entre el color deseado y cada color de la paleta es calculada como si los valores RGB representasen puntos en un espacio tridimensional.

Vea también **imagecolorexact()**.

imagecolorclosestalpha

(PHP 4 >= 4.0.6, PHP 5)

imagecolorclosestalpha -- Get the index of the closest color to the specified color + alpha

Description

int **imagecolorclosestalpha** (resource image, int red, int green, int blue, int alpha)

Returns the index of the color in the palette of the image which is "closest" to the specified RGB value and *alpha* level.

> **Nota:** Esta funcion requiere GD 2.0.1 o posterior.

See also **imagecolorexactalpha()**.

imagecolorclosesthwb

(PHP 4 >= 4.0.1, PHP 5)

imagecolorclosesthwb -- Get the index of the color which has the hue, white and blackness nearest to the given color

Description

int **imagecolorclosesthwb** (resource image, int red, int green, int blue)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

imagecolordeallocate

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

imagecolordeallocate -- De-allocate a color for an image

Description

int **imagecolordeallocate** (resource image, int color)

The **imagecolordeallocate()** function de-allocates a color previously allocated with **imagecolorallocate()** or **imagecolorallocatealpha()**.

```
<?php
$white = imagecolorallocate($im, 255, 255, 255);
imagecolordeallocate($im, $white);
?>
```

See also **imagecolorallocate()** and **imagecolorallocatealpha()**.

ImageColorExact

(PHP 3, PHP 4 , PHP 5)

ImageColorExact -- Devuelve el índice del color especificado

Descripción

int **imagecolorexact** (int im, int red, int green, int blue)

Devuelve el índice del color especificado de la paleta de la imagen.

Si el color no existe en la paleta de la imagen, se devuelve el valor -1.

Vea también **imagecolorclosest()**.

imagecolorexactalpha

(PHP 4 >= 4.0.6, PHP 5)

imagecolorexactalpha -- Get the index of the specified color + alpha

Description

int **imagecolorexactalpha** (resource image, int red, int green, int blue, int alpha)

Returns the index of the specified color+alpha in the palette of the image.

If the color does not exist in the image's palette, -1 is returned.

> **Nota:** Esta funcion requiere GD 2.0.1 o posterior.

See also **imagecolorclosestalpha()**.

imagecolormatch

(PHP 4 >= 4.3.0, PHP 5)

imagecolormatch -- Makes the colors of the palette version of an image more closely match the true color version

Description

bool **imagecolormatch** (resource image1, resource image2)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

image1 must be Truecolor, *image2* must be Palette, and both *image1* and *image2* must be the same size.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

> **Nota:** Esta funcion esta disponible solamente si PHP se ha compilado con la version de las bibliotecas GD distribuidas con PHP.

> **Nota:** Esta funcion requiere GD 2.0.1 o posterior.

See also **imagecreatetruecolor()**.

ImageColorResolve

(PHP 3>= 3.0.2, PHP 4 , PHP 5)

ImageColorResolve -- Devuelve el índice del color especificado o su posible alternativa más cercana

Descripción

int **imagecolorresolve** (int im, int red, int green, int blue)

Esta función garantiza el resultado de un índice de color para un color solicitado, ya sea el color exacto o su alternativa más cercana.

Vea también **imagecolorclosest()**.

imagecolorresolvealpha

(PHP 4 >= 4.0.6, PHP 5)

imagecolorresolvealpha -- Get the index of the specified color + alpha or its closest possible alternative

Description

int **imagecolorresolvealpha** (resource image, int red, int green, int blue, int alpha)

This function is guaranteed to return a color index for a requested color, either the exact color or the

closest possible alternative.

Nota: Esta funcion requiere GD 2.0.1 o posterior.

See also **imagecolorclosestalpha()**.

ImageColorSet

(PHP 3, PHP 4 , PHP 5)

ImageColorSet -- Establece el color para el índice de la paleta especificado

Descripción

bool **imagecolorset** (int im, int index, int red, int green, int blue)

Establece el índice especificado de la paleta con el color introduciod. Esto es útil para crear efectos de rellenado en imagenes con paletas sin la sobrecarga de tener que realizar el rellenado.

Vea también **imagecolorat()**.

ImageColorsForIndex

(PHP 3, PHP 4 , PHP 5)

ImageColorsForIndex -- Obtiene los colores de un índice

Descripción

array **imagecolorsforindex** (int im, int index)

Devuelve una matriz asociativa con las claves red, green y blue que contienen los valores apropiados para el color especificado en el índice.

Vea también **imagecolorat()** y **imagecolorexact()**.

ImageColorsTotal

(PHP 3, PHP 4 , PHP 5)

ImageColorsTotal -- Encuentra el número de colores de la paleta de una imagen

Descripción

int **imagecolorstotal** (int im)

Encuentra el número de colores de la paleta de una imagen.

Vea también **imagecolorat()** y **imagecolorsforindex()**.

ImageColorTransparent

(PHP 3, PHP 4 , PHP 5)

ImageColorTransparent -- Define un color como transparente

Descripción

int **imagecolortransparent** (int im [, int col])

ImageColorTransparent establece como color transparente a col en la imagen im. im es el identificador de imagen devuelto por **imagecreate()** y col es el identificador de color devuelto por **imagecolorallocate()**.

Se devuelve el identificador del color transparente (o el actual, si no se especifica ninguno).

imagecopy

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

imagecopy -- Copy part of an image

Description

int **imagecopy** (resource dst_im, resource src_im, int dst_x, int dst_y, int src_x, int src_y, int src_w, int src_h)

Copy a part of *src_im* onto *dst_im* starting at the x,y coordinates *src_x*, *src_y* with a width of *src_w* and a height of *src_h*. The portion defined will be copied onto the x,y coordinates, *dst_x* and *dst_y*.

imagecopymerge

(PHP 4 >= 4.0.1, PHP 5)

imagecopymerge -- Copy and merge part of an image

Description

int **imagecopymerge** (resource dst_im, resource src_im, int dst_x, int dst_y, int src_x, int src_y, int src_w, int src_h, int pct)

Copy a part of *src_im* onto *dst_im* starting at the x,y coordinates *src_x*, *src_y* with a width of *src_w* and a height of *src_h*. The portion defined will be copied onto the x,y coordinates, *dst_x* and *dst_y*. The two images will be merged according to *pct* which can range from 0 to 100. When *pct* = 0, no action is taken, when 100 this function behaves identically to **imagecopy()** for pallete images, while it implements alpha transparency for true colour images.

> **Nota:** This function was added in PHP 4.0.6

imagecopymergegray

(PHP 4 >= 4.0.6, PHP 5)

imagecopymergegray -- Copy and merge part of an image with gray scale

Description

int **imagecopymergegray** (resource dst_im, resource src_im, int dst_x, int dst_y, int src_x, int src_y, int src_w, int src_h, int pct)

imagecopymergegray() copy a part of *src_im* onto *dst_im* starting at the x,y coordinates *src_x*, *src_y* with a width of *src_w* and a height of *src_h*. The portion defined will be copied onto the x,y coordinates, *dst_x* and *dst_y*. The two images will be merged according to *pct* which can range from 0 to 100. When *pct* = 0, no action is taken, when 100 this function behaves identically to **imagecopy()**.

This function is identical to **imagecopymerge()** except that when merging it preserves the hue of the source by converting the destination pixels to gray scale before the copy operation.

> **Nota:** This function was added in PHP 4.0.6

imagecopyresampled

(PHP 4 >= 4.0.6, PHP 5)

imagecopyresampled -- Copia y reescala parte de una imagen con remuestreo

Descripción

int **imagecopyresampled** (resource img_dst, resource img_org, int Xdst, int Ydst, int Xorg, int Yorg, int ancho_dst, int alto_dst, int ancho_org, int alto_org)

imagecopyresampled() copia una porción rectangular de una imagen sobre otra, suavizando los valores de los píxeles mediante interpolación, de forma que al reducir el tamaño de una imagen aún mantiene una buena claridad. *img_dst* es la imagen de destino, *img_org* es el identificador de la imagen de origen. Si las coordenadas de origen y destino y ancho y alto son diferentes, se encogerá o agrandará el fragmento de imagen según sea necesario. Las coordenadas son relativas a la esquina superior izquierda. Esta función se puede usar para copiar regiones dentro de la misma imagen (si *img_dst* es la misma que *img_org*) pero si las regiones se superponen, los resultados serán impredecibles.

Vea también **imagecopyresized()**.

> **Nota:** Esta función fue añadida en PHP 4.0.6 y requiere GD 2.0.1 o superior

ImageCopyResized

(PHP 3, PHP 4 , PHP 5)

ImageCopyResized -- Copia y redimensiona parte de una imagen

Descripción

int **imagecopyresized** (int dst_im, int src_im, int dstX, int dstY, int srcX, int srcY, int dstW, int dstH, int srcW, int srcH)

ImageCopyResize copia una porción rectangular de una imagen hacia otra imagen. *dst_im* es la imagen de destino, *src_im* es el identificador de la imagen origen. Si la altura y anchura de las coordenadas de origen y destino difieren se realizará un estrechamiento o un estiramiento apropiado del fragmento de la imagen. Las coordenadas van localizadas sobre la esquina superior izquierda. Esta función se puede usar para copiar regiones dentro de la misma imagen (si *dst_im* es igual que *src_im*) pero si las regiones se solapan los resultados seran impredecibles.

ImageCreate

(PHP 3, PHP 4 , PHP 5)

ImageCreate -- Crea una nueva imagen

Descripción

int **imagecreate** (int x_size, int y_size)

ImageCreate devuelve un identificador de imagen representando una imagen en blanco de tamaño x_size por y_size.

imagecreatefromgd2

(PHP 4 >= 4.1.0, PHP 5)

imagecreatefromgd2 -- Create a new image from GD2 file or URL

Description

resource **imagecreatefromgd2** (string filename)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

Nota: Esta funcion requiere GD 2.0.1 o posterior.

Sugerencia: Puede usar una URL como nombre de archivo con esta función si los fopen wrappers han sido activados. Consulte **fopen()** para más detalles sobre cómo especificar el nombre de fichero y Apéndice L una lista de protocolos URL soportados

Aviso
Versiones de *PHP* para Windows anteriores a 4.3.0, no soportan el acceso remoto a archivos para esta función, no funcionará ni activando siquiera allow_url_fopen.

imagecreatefromgd2part

(PHP 4 >= 4.1.0, PHP 5)

imagecreatefromgd2part -- Create a new image from a given part of GD2 file or URL

Description

resource **imagecreatefromgd2part** (string filename, int srcX, int srcY, int width, int height)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

Nota: Esta funcion requiere GD 2.0.1 o posterior.

Sugerencia: Puede usar una URL como nombre de archivo con esta función si los fopen wrappers han sido activados. Consulte **fopen()** para más detalles sobre cómo especificar el nombre de fichero y Apéndice L una lista de protocolos URL soportados

Aviso
Versiones de *PHP* para Windows anteriores a 4.3.0, no soportan el acceso remoto a archivos para esta función, no funcionará ni activando siquiera allow_url_fopen.

imagecreatefromgd

(PHP 4 >= 4.1.0, PHP 5)

imagecreatefromgd -- Create a new image from GD file or URL

Description

resource **imagecreatefromgd** (string filename)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

Sugerencia: Puede usar una URL como nombre de archivo con esta función si los fopen wrappers han sido activados. Consulte **fopen()** para más detalles sobre cómo especificar el nombre de fichero y Apéndice L una lista de protocolos URL soportados

Aviso
Versiones de *PHP* para Windows anteriores a 4.3.0, no soportan el acceso remoto a archivos para esta función, no funcionará ni activando siquiera allow_url_fopen.

imagecreatefromgif

(PHP 3, PHP 4 , PHP 5)

imagecreatefromgif -- Crear una nueva imagen a partir de un archivo o URL

Descripción

resource **imagecreatefromgif** (string nombre_archivo)

imagecreatefromgif() devuelve un identificador de imagen que representa la imagen obtenida desde el nombre de archivo dado.

imagecreatefromgif() devuelve una cadena vacía en caso de fallo. También genera un mensaje de error, el cual, desafortunadamente, se despliega como un enlace roto en un navegador. Para facilitar la depuración, el siguiente ejemplo producirá un GIF de error:
Ejemplo 1. Ejemplo para gestionar un error duranto la creación (cortesía de vic arroba zymsys punto com)

```
<?php
function LoadGif ($nombre_imagen)
{
    $im = @imagecreatefromgif ($nombre_imagen); /* Intentar la apertura */
    if (!$im) { /* Verificar si ha fallado */
        $im = imagecreate (150, 30); /* Crear una imagen en blanco */
        $bgc = imagecolorallocate ($im, 255, 255, 255);
        $tc = imagecolorallocate ($im, 0, 0, 0);
        imagefilledrectangle ($im, 0, 0, 150, 30, $bgc);
        /* Generar un mensaje de error */
        imagestring ($im, 1, 5, 5, "Error en la carga de $nombre_imagen", $tc);
    }
    return $im;
}
?>
```

> **Nota:** Dado que el soporte de GIF fue completamente retirado de la biblioteca GD en la versión 1.6, esta función no se encuentra disponible si está usando tal versión de la biblioteca GD.
>
> **Sugerencia:** Puede usar una URL como nombre de archivo con esta función si los fopen wrappers han sido activados. Consulte **fopen()** para más detalles sobre cómo especificar el nombre de fichero y Apéndice L una lista de protocolos URL soportados

Aviso
Versiones de *PHP* para Windows anteriores a 4.3.0, no soportan el acceso remoto a archivos para esta función, no funcionará ni activando siquiera allow_url_fopen.

imagecreatefromjpeg

(PHP 3>= 3.0.16, PHP 4 , PHP 5)

imagecreatefromjpeg -- Crea una imagen nueva desde un archivo o URL

Descripción

resource **imagecreatefromjpeg** (string nombre_archivo)

imagecreatefromjpeg() devuelve un identificador de imagen que representa a la imagen obtenida a partir del nombre de archivo indicado.

imagecreatefromjpeg() devuelve una cadena vacía si ha fallado. También escribe un mensaje de error, que desafortunadamente se muestra en el navegador como un enlace roto. Para depurar con mayor comodidad, el ejemplo siguiente producirá un JPEG erróneo:

Ejemplo 1. Ejemplo de cómo manipular un error durante la creación (cortesía de vic@zymsys.com)

```
function CargarJpeg ($nombreimg) {
    $im = @imagecreatefromjpeg ($nombreimg); /* Intento de apertura */
    if (!$im) { /* Comprobar si ha fallado */
        $im  = imagecreate (150, 30); /* Crear una imagen en blanco */
        $bgc = imagecolorallocate ($im, 255, 255, 255);
        $tc  = imagecolorallocate ($im, 0, 0, 0);
        imagefilledrectangle ($im, 0, 0, 150, 30, $bgc);
        /* Mostrar un mensaje de error */
        imagestring ($im, 1, 5, 5, "Error cargando $nombreimg", $tc);
    }
    return $im;
}
```

imagecreatefrompng

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

imagecreatefrompng -- Crea una imagen nueva desde un archivo o URL

Descripción

resource **imagecreatefrompng** (string nombre_archivo)

imagecreatefrompng() devuelve un identificador de imagen que representa a la imagen obtenida a partir del nombre de archivo indicado.

imagecreatefrompng() devuelve una cadena vacía si ha fallado. También escribe un mensaje de error, que desafortunadamente se muestra en el navegador como un enlace roto. Para depurar con mayor comodidad, el ejemplo siguiente producirá un JPEG erróneo:

Ejemplo 1. Ejemplo de cómo manipular un error durante la creación (cortesía de vic@zymsys.com)

```
function CargarJpeg ($nombreimg) {
    $im = @imagecreatefrompng ($nombreimg); /* Intento de apertura */
    if (!$im) { /* Comprobar si ha fallado */
        $im  = imagecreate (150, 30); /* Crear una imagen en blanco */
        $bgc = imagecolorallocate ($im, 255, 255, 255);
        $tc  = imagecolorallocate ($im, 0, 0, 0);
        imagefilledrectangle ($im, 0, 0, 150, 30, $bgc);
        /* Mostrar un mensaje de error */
        imagestring ($im, 1, 5, 5, "Error cargando $nombreimg", $tc);
    }
    return $im;
}
```

imagecreatefromstring

(PHP 4 >= 4.0.4, PHP 5)

imagecreatefromstring -- Create a new image from the image stream in the string

Descripción

resource **imagecreatefromstring** (string image)

imagecreatefromstring() returns an image identifier representing the image obtained from the given string. These types will be automatically detected if your build of PHP supports them: JPEG, PNG, GIF, WBMP, and GD2.

Valores retornados

An image resource will be returned on success. **FALSE** is returned if the image type is unsupported, the data is not in a recognised format, or the image is corrupt and cannot be loaded.

Ver también

imagecreatefromjpeg(), **imagecreatefrompng()**, **imagecreatefromgif()**, y **imagecreate()**

Ejemplos

Ejemplo 1. imagecreatefromstring() example

```
<?php
$data = 'iVBORw0KGgoAAAANSUhEUgAAABwAAAASCAMAAAB/2U7WAAAABl'
       . 'BMVEUAAAD///+l2Z/dAAAASUlEQVR4XqWQUQoAIAxC2/0vXZDr'
       . 'EX4IJTRkb7lobNUStXsB0jIXIAMSsQnWlsV+wULF4Avk9fLq2r'
       . '8a5HSE35Q3eO2XP1A1wQkZSgETvDtKdQAAAABJRU5ErkJggg==';
$data = base64_decode($data);

$im = imagecreatefromstring($data);
if ($im !== false) {
    header('Content-Type: image/png');
    imagepng($im);
}
else {
    echo 'An error occured.';
}
?>
```

imagecreatefromwbmp

(PHP 4 >= 4.0.1, PHP 5)

imagecreatefromwbmp -- Create a new image from file or URL

Description

resource **imagecreatefromwbmp** (string filename)

imagecreatefromwbmp() returns an image identifier representing the image obtained from the given filename.

imagecreatefromwbmp() returns an empty string on failure. It also outputs an error message, which unfortunately displays as a broken link in a browser. To ease debugging the following example will produce an error WBMP:

Ejemplo 1. Example to handle an error during creation (courtesy vic at zymsys dot com)

```php
<?php
function LoadWBMP($imgname)
{
    $im = @imagecreatefromwbmp($imgname); /* Attempt to open */
    if (!$im) { /* See if it failed */
        $im  = imagecreate (20, 20); /* Create a blank image */
        $bgc = imagecolorallocate($im, 255, 255, 255);
        $tc  = imagecolorallocate($im, 0, 0, 0);
        imagefilledrectangle($im, 0, 0, 10, 10, $bgc);
        /* Output an errmsg */
        imagestring($im, 1, 5, 5, "Error loading $imgname", $tc);
    }
    return $im;
}
?>
```

Nota: WBMP support is only available if PHP was compiled against GD-1.8 or later.

Sugerencia: Puede usar una URL como nombre de archivo con esta función si los fopen wrappers han sido activados. Consulte **fopen()** para más detalles sobre cómo especificar el nombre de fichero y Apéndice L una lista de protocolos URL soportados

Aviso
Versiones de *PHP* para Windows anteriores a 4.3.0, no soportan el acceso remoto a archivos para esta función, no funcionará ni activando siquiera allow_url_fopen.

imagecreatefromxbm

(PHP 4 >= 4.0.1, PHP 5)

imagecreatefromxbm -- Create a new image from file or URL

Description

resource **imagecreatefromxbm** (string filename)

imagecreatefromxbm() returns an image identifier representing the image obtained from the given filename.

Sugerencia: Puede usar una URL como nombre de archivo con esta función si los fopen wrappers han sido activados. Consulte **fopen()** para más detalles sobre cómo especificar el nombre de fichero y Apéndice L una lista de protocolos URL soportados

Aviso
Versiones de *PHP* para Windows anteriores a 4.3.0, no soportan el acceso remoto a archivos para esta función, no funcionará ni activando siquiera allow_url_fopen.

imagecreatefromxpm

(PHP 4 >= 4.0.1, PHP 5)

imagecreatefromxpm -- Create a new image from file or URL

Description

resource **imagecreatefromxpm** (string filename)

imagecreatefromxpm() returns an image identifier representing the image obtained from the given filename.

> **Nota:** Esta funcion esta disponible solamente si PHP se ha compilado con la version de las bibliotecas GD distribuidas con PHP.

> **Sugerencia:** Puede usar una URL como nombre de archivo con esta función si los fopen wrappers han sido activados. Consulte **fopen()** para más detalles sobre cómo especificar el nombre de fichero y Apéndice L una lista de protocolos URL soportados

Aviso
Versiones de *PHP* para Windows anteriores a 4.3.0, no soportan el acceso remoto a archivos para esta función, no funcionará ni activando siquiera allow_url_fopen.

imagecreatetruecolor

(PHP 4 >= 4.0.6, PHP 5)

imagecreatetruecolor -- Crea una imagen nueva en color real (true color)

Descripción

resource **imagecreatetruecolor** (int anchura, int altura)

imagecreatetruecolor() devuelve un identificador de imagen representando una imagen en blanco de tamaño *anchura* por *altura*.

> **Nota:** Esta función fue añadida en PHP 4.0.6 y requiere GD 2.0.1 o superior

ImageDashedLine

(PHP 3, PHP 4 , PHP 5)

ImageDashedLine -- Dibuja una línea discontinua

Descripción

int **imagedashedline** (int im, int x1, int y1, int x2, int y2, int col)

ImageLine dibuja una línea discontinua desde x1,y1 hasta x2, y2 (arriba izquierda es 0.0) en la imagen im con el color col.

Vea también **imageline()**.

ImageDestroy

(PHP 3, PHP 4 , PHP 5)

ImageDestroy -- Destruye una imagen

Descripción

int **imagedestroy** (int im)

ImageDestroy libera la memoria asociada a la imagen im. im es la imagen devuelta por la función **imagecreate()**.

imageellipse

(PHP 4 >= 4.0.6, PHP 5)

imageellipse -- Draw an ellipse

Description

int **imageellipse** (resource image, int cx, int cy, int w, int h, int color)

imageellipse() draws an ellipse centered at *cx*, *cy* (top left is 0, 0) in the image represented by *image*. *W* and *h* specifies the ellipse's width and height respectively. The color of the ellipse is specified by *color*.

> **Nota:** This function was added in PHP 4.0.6 and requires GD 2.0.2 or later which can be obtained at http://www.boutell.com/gd/

Ejemplo 1. imageellipse() example

```
<?php

// create a blank image
$image = imagecreate(400, 300);

// fill the background color
$bg = imagecolorallocate($image, 0, 0, 0);

// choose a color for the ellipse
$col_ellipse = imagecolorallocate($image, 255, 255, 255);

// draw the ellipse
imageellipse($image, 200, 150, 300, 200, $col_ellipse);

// output the picture
header("Content-type: image/png");
imagepng($image);

?>
```

See also **imagefilledellipse()** and **imagearc()**.

ImageFill

(PHP 3, PHP 4 , PHP 5)

ImageFill -- Relleno

Descripción

int **imagefill** (int im, int x, int y, int col)

ImageFill realiza un relleno empezando en la coordenada x,y (arriba izquierda es 0,0) con el color col en la imagen im.

imagefilledarc

(PHP 4 >= 4.0.6, PHP 5)

imagefilledarc -- Draw a partial ellipse and fill it

Description

bool **imagefilledarc** (resource image, int cx, int cy, int w, int h, int s, int e, int color, int style)

imagefilledarc() draws a partial ellipse centered at *cx*, *cy* (top left is 0, 0) in the image represented by *image*. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo. *W* and *h* specifies the ellipse's width and height respectively while the start and end points are specified in degrees indicated by the *s* and *e* arguments. *style* is a bitwise OR of the following possibilities:

1. **IMG_ARC_PIE**

2. **IMG_ARC_CHORD**

3. **IMG_ARC_NOFILL**

4. **IMG_ARC_EDGED**

IMG_ARC_PIE and **IMG_ARC_CHORD** are mutually exclusive; **IMG_ARC_CHORD** just connects the starting and ending angles with a straight line, while **IMG_ARC_PIE** produces a rounded edge. **IMG_ARC_NOFILL** indicates that the arc or chord should be outlined, not filled. **IMG_ARC_EDGED**, used together with **IMG_ARC_NOFILL**, indicates that the beginning and ending angles should be connected to the center - this is a good way to outline (rather than fill) a 'pie slice'.

Ejemplo 1. Creating a 3D looking pie

```
<?php

// this example is provided by poxy at klam dot is

// create image
$image = imagecreate(100, 100);

// allocate some solors
$white    = imagecolorallocate($image, 0xFF, 0xFF, 0xFF);
$gray     = imagecolorallocate($image, 0xC0, 0xC0, 0xC0);
$darkgray = imagecolorallocate($image, 0x90, 0x90, 0x90);
$navy     = imagecolorallocate($image, 0x00, 0x00, 0x80);
$darknavy = imagecolorallocate($image, 0x00, 0x00, 0x50);
$red      = imagecolorallocate($image, 0xFF, 0x00, 0x00);
$darkred  = imagecolorallocate($image, 0x90, 0x00, 0x00);

// make the 3D effect
for ($i = 60; $i > 50; $i--) {
   imagefilledarc($image, 50, $i, 100, 50, 0, 45, $darknavy, IMG_ARC_PIE);
  imagefilledarc($image, 50, $i, 100, 50, 45, 75 , $darkgray, IMG_ARC_PIE);
  imagefilledarc($image, 50, $i, 100, 50, 75, 360 , $darkred, IMG_ARC_PIE);
}

imagefilledarc($image, 50, 50, 100, 50, 0, 45, $navy, IMG_ARC_PIE);
imagefilledarc($image, 50, 50, 100, 50, 45, 75 , $gray, IMG_ARC_PIE);
imagefilledarc($image, 50, 50, 100, 50, 75, 360 , $red, IMG_ARC_PIE);


// flush image
header('Content-type: image/png');
imagepng($image);
imagedestroy($image);
?>
```

Nota: Esta funcion requiere GD 2.0.1 o posterior.

imagefilledellipse

(PHP 4 >= 4.0.6, PHP 5)

imagefilledellipse -- Draw a filled ellipse

Description

bool **imagefilledellipse** (resource image, int cx, int cy, int w, int h, int color)

imagefilledellipse() draws an ellipse centered at *cx*, *cy* (top left is 0, 0) in the image represented by *image*. *W* and *h* specifies the ellipse's width and height respectively. The ellipse is filled using *color*. Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Nota: This function was added in PHP 4.0.6 and requires GD 2.0.1 or later

Ejemplo 1. imagefilledellipse() example

```php
<?php

// create a blank image
$image = imagecreate(400, 300);

// fill the background color
$bg = imagecolorallocate($image, 0, 0, 0);

// choose a color for the ellipse
$col_ellipse = imagecolorallocate($image, 255, 255, 255);

// draw the white ellipse
imagefilledellipse($image, 200, 150, 300, 200, $col_ellipse);

// output the picture
header("Content-type: image/png");
imagepng($image);

?>
```

Nota: Esta funcion requiere GD 2.0.1 o posterior.

See also **imageellipse()** and **imagefilledarc()**.

ImageFilledPolygon

(PHP 3, PHP 4 , PHP 5)

ImageFilledPolygon -- Dibuja un polígono relleno

Descripción

int **imagefilledpolygon** (int im, array points, int num_points, int col)

ImageFilledPolygon crea un polígono relleno en la imagen im, points es una matriz PHP conteniendo los vértices del poligono, por ejemplo. points[0] = x0, points[1] = y0, points[2] = x1, points[3] = y1, etc. num_points es el número total de vértices.

ImageFilledRectangle

(PHP 3, PHP 4 , PHP 5)

ImageFilledRectangle -- dibuja un rectángulo relleno

Descripción

int **imagefilledrectangle** (int im, int x1, int y1, int x2, int y2, int col)

ImageFilledRectangle crea un rectángulo relleno con color col en la imagen im comenzando con las coordenadas superiores izquierdas x1, y1 y finalizando en las coordenadas inferiores derechas x2, y2. 0,0 es la esquina superior izquierda de la imagen.

ImageFillToBorder

(PHP 3, PHP 4 , PHP 5)

ImageFillToBorder -- Relleno de un color especifico

Descripción

int **imagefilltoborder** (int im, int x, int y, int border, int col)

ImageFillToBorder realiza un relleno hasta el color del borde que está definido por border. El punto de inicio para el relleno es x,y (arriba izquierda es 0,0) y la región se rellena con el color col.

imagefilter

(PHP 5)

imagefilter -- Applies a filter to an image

Description

bool **imagefilter** (resource src_im, int filtertype [, int arg1 [, int arg2 [, int arg3]]])

imagefilter() applies the filter *filtertype* to the image, using *arg1*, *arg2* and *arg3* where necessary.

filtertype can be one of the following:

- *IMG_FILTER_NEGATE*: Reverses all colors of the image.
- *IMG_FILTER_GRAYSCALE*: Converts the image into grayscale.
- *IMG_FILTER_BRIGHTNESS*: Changes the brightness of the image. Use *arg1* to set the level of brightness.
- *IMG_FILTER_CONTRAST*: Changes the contrast of the image. Use *arg1* to set the level of contrast.
- *IMG_FILTER_COLORIZE*: Like *IMG_FILTER_GRAYSCALE*, except you can specify the color. Use *arg1*, *arg2* and *arg3* in the form of *red*, *blue*, *green*. The range for each color is 0 to 255.
- *IMG_FILTER_EDGEDETECT*: Uses edge detection to highlight the edges in the image.
- *IMG_FILTER_EMBOSS*: Embosses the image.
- *IMG_FILTER_GAUSSIAN_BLUR*: Blurs the image using the Gaussian method.
- *IMG_FILTER_SELECTIVE_BLUR*: Blurs the image.
- *IMG_FILTER_MEAN_REMOVAL*: Uses mean removal to achieve a "sketchy" effect.
- *IMG_FILTER_SMOOTH*: Makes the image smoother. Use *arg1* to set the level of smoothness.

Nota: Esta funcion esta disponible solamente si PHP se ha compilado con la version de las bibliotecas GD distribuidas con PHP.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplo 1. imagefilter() grayscale example

```
<?php
$im = imagecreatefrompng('dave.png');
if ($im && imagefilter($im, IMG_FILTER_GRAYSCALE)) {
    echo 'Image converted to grayscale.';
    imagepng($im, 'dave.png');
} else {
    echo 'Conversion to grayscale failed.';
}

imagedestroy($im);
?>
```

Ejemplo 2. imagefilter() brightness example

```
<?php
$im = imagecreatefrompng('sean.png');
if ($im && imagefilter($im, IMG_FILTER_BRIGHTNESS, 20)) {
    echo 'Image brightness changed.';
    imagepng($im, 'sean.png');
} else {
    echo 'Image brightness change failed.';
}

imagedestroy($im);
?>
```

Ejemplo 3. imagefilter() colorize example

```
<?php
$im = imagecreatefrompng('philip.png');

/* R, G, B, so 0, 255, 0 is green */
if ($im && imagefilter($im, IMG_FILTER_COLORIZE, 0, 255, 0)) {
    echo 'Image successfully shaded green.';
    imagepng($im, 'philip.png');
} else {
    echo 'Green shading failed.';
}

imagedestroy($im);
?>
```

ImageFontHeight

(PHP 3, PHP 4 , PHP 5)

ImageFontHeight -- Devuelve la altura de una fuente

Descripción

int **imagefontheight** (int font)

Devuelve la altura en pixels de un carácter en un fuente específica.

Vea también **imagefontwidth()** y **imageloadfont()**.

ImageFontWidth

(PHP 3, PHP 4 , PHP 5)

ImageFontWidth -- Devuelve la anchura de una fuente

Descripción

int **imagefontwidth** (int font)

Devuelve la anchura en pixels de un carácter en un fuente específica.

Vea también **imagefontheight()** y **imageloadfont()**.

imageftbbox

(PHP 4 >= 4.1.0, PHP 5)

imageftbbox -- Give the bounding box of a text using fonts via freetype2

Description

array **imageftbbox** (float size, float angle, string font_file, string text [, array extrainfo])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

Nota: Esta funcion requiere GD 2.0.1 o posterior.

Nota: Parameter *extrainfo* is optional since PHP 4.3.5.

imagefttext

(PHP 4 >= 4.1.0, PHP 5)

imagefttext -- Write text to the image using fonts using FreeType 2

Description

array **imagefttext** (resource image, float size, float angle, int x, int y, int col, string font_file, string text [, array extrainfo])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

Nota: Esta funcion requiere GD 2.0.1 o posterior.

Nota: Parameter *extrainfo* is optional since PHP 4.3.5.

imagegammacorrect

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

imagegammacorrect -- Apply a gamma correction to a GD image

Description

int **imagegammacorrect** (resource image, float inputgamma, float outputgamma)

The **imagegammacorrect()** function applies gamma correction to a gd image stream (*image*) given an input gamma, the parameter *inputgamma* and an output gamma, the parameter *outputgamma*.

imagegd2

(PHP 4 >= 4.1.0, PHP 5)

imagegd2 -- Output GD2 image to browser or file

Description

bool **imagegd2** (resource image [, string filename [, int chunk_size [, int type]]])

imagegd2() outputs a GD2 image to *filename*. The *image* parameter is the return from the **imagecreate()** function.

The *filename* parameter is optional, and if left off, the raw image stream will be output directly.

The optional *type* parameter is either **`IMG_GD2_RAW`** or **`IMG_GD2_COMPRESSED`**. Default is **`IMG_GD2_RAW`**.

> **Nota:** The optional *chunk_size* and *type* parameters became available in PHP 4.3.2.
>
> **Nota:** The GD2 format is commonly used to allow fast loading of parts of images. Note that the GD2 format is only usable in GD2-compatible applications.
>
> **Nota:** Esta funcion requiere GD 2.0.1 o posterior.

See also **imagegd()**

imagegd

(PHP 4 >= 4.1.0, PHP 5)

imagegd -- Output GD image to browser or file

Description

bool **imagegd** (resource image [, string filename])

imagegd() outputs a GD image to *filename*. The *image* argument is the return from the **imagecreate()** function.

The *filename* parameter is optional, and if left off, the raw image stream will be output directly.

Nota: The GD format is commonly used to allow fast loading of parts of images. Note that the GD format is only usable in GD-compatible applications.

See also **imagegd2()**.

imagegif

(PHP 3, PHP 4 , PHP 5)

imagegif -- Producir la salida de una imagen al navegador o a un archivo

Descripción

bool **imagegif** (resource imagen [, string nombre_archivo])

imagegif() crea el archivo GIF indicado por nombre_archivo desde la imagen *imagen*. El parámetro *imagen* es el valor de retorno de la función **imagecreate()**.

El formato de la imagen será GIF87a a menos que la imagen haya sido convertida a transparente con **imagecolortransparent()**, en cuyo caso el formato de la imagen será GIF89a.

El argumento nombre_archivo es opcional, y si se omite, la secuencia cruda de la imagen será dirigida a la salida estándar directamente. Al enviar un valor content-type image/gif mediante **header()**, puede crear un script PHP que genere imágenes GIF directamente.

Nota: Dado que el soporte GIF fue retirado por completo de la biblioteca GD en la versión 1.6, ésta función no se encuentra disponible si está usando tal versión de la biblioteca GD. Se espera que el soporte para esta característica regrese en una versión posterior al relanzamiento del soporte GIF en la biblioteca GD a mediados de 2004. Para más información, consulte el sitio web del Proyecto GD.

El siguiente segmento de código le permite escribir aplicaciones PHP más portables al detectar automáticamente el tipo de soporte GD que se encuentra disponible. Reemplace la secuencia *header ("Content-type: image/gif"); imagegif ($im);* por la secuencia más flexible:

```
<?php
if (function_exists("imagegif")) {
    header("Content-type: image/gif");
    imagegif($im);
} elseif (function_exists("imagejpeg")) {
    header("Content-type: image/jpeg");
    imagejpeg($im, "", 0.5);
} elseif (function_exists("imagepng")) {
    header("Content-type: image/png");
    imagepng($im);
} elseif (function_exists("imagewbmp")) {
    header("Content-type: image/vnd.wap.wbmp");
    imagewbmp($im);
} else {
    die("No hay soporte de im&aacute;genes en este servidor PHP");
}
?>
```

Nota: A partir de la versión 3.0.18 y 4.0.2 puede usar la función **imagetypes()** en lugar de **function_exists()** para chequear por la presencia de los varios formatos de imagen soportados:

```
<?php
if (imagetypes() & IMG_GIF) {
    header ("Content-type: image/gif");
    imagegif ($im);
} elseif (imagetypes() & IMG_JPG) {
    /* ... etc. */
}
?>
```

Vea también **imagepng()**, **imagewbmp()**, **imagejpeg()** y **imagetypes()**.

ImageInterlace

(PHP 3, PHP 4 , PHP 5)

ImageInterlace -- Activa o desactiva el entrelazado

Descripción

int **imageinterlace** (int im [, int interlace])

ImageInterlace() activa o desactiva el bit de entrelazado. Si interlace es 1 la imagen im será entrelazada, y si interlace es 0 el bit de entrelazado se desactiva.

Esta función devuelve como ha cambiado el estado del bit de entralazado de la imagen.

imageistruecolor

(PHP 4 >= 4.3.2, PHP 5)

imageistruecolor -- Finds whether an image is a truecolor image

Description

bool **imageistruecolor** (resource image)

imageistruecolor() finds whether the image *image* is a truecolor image.

> **Nota:** Esta funcion requiere GD 2.0.1 o posterior.

See also **imagecreatetruecolor()**.

imagejpeg

(PHP 3>= 3.0.16, PHP 4 , PHP 5)

imagejpeg -- Output image to browser or file

Description

bool **imagejpeg** (resource image [, string filename [, int quality]])

imagejpeg() creates the JPEG file in filename from the image *image*. The *image* argument is the return from the **imagecreate()** function.

The filename argument is optional, and if left off, the raw image stream will be output directly. To skip the filename argument in order to provide a quality argument just use an empty string ('). By sending an image/jpeg content-type using **header()**, you can create a PHP script that outputs JPEG images directly.

> **Nota:** JPEG support is only available if PHP was compiled against GD-1.8 or later.

quality is optional, and ranges from 0 (worst quality, smaller file) to 100 (best quality, biggest file). The default is the default IJG quality value (about 75).

If you want to output Progressive JPEGs, you need to set interlacing on with **imageinterlace()**.

See also **imagepng()**, **imagegif()**, **imagewbmp()**, **imageinterlace()** and **imagetypes()**.

imagelayereffect

(PHP 4 >= 4.3.0, PHP 5)

imagelayereffect -- Set the alpha blending flag to use the bundled libgd layering effects

Description

bool **imagelayereffect** (resource image, int effect)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

> **Nota:** Esta funcion esta disponible solamente si PHP se ha compilado con la version de las bibliotecas GD distribuidas con PHP.

> **Nota:** Esta funcion requiere GD 2.0.1 o posterior.

ImageLine

(PHP 3, PHP 4 , PHP 5)

ImageLine -- Dibuja una línea

Descripción

int **imageline** (int im, int x1, int y1, int x2, int y2, int col)

ImageLine dibuja una línea desde x1,y1 hasta x2,y2 (arriba izquierda es 0,0) en la imagen im con el color col.

Vea también **imagecreate()** y **imagecolorallocate()**.

imageloadfont

(PHP 3, PHP 4 , PHP 5)

imageloadfont -- Cargar una fuente nueva

Descripción

int **imageloadfont** (string archivo)

imageloadfont() carga una fuente de bitmaps definida por el usuario y devuelve un identificador para la fuente (que siempre es mayor de 5, de modo que no entre en conflicto con las fuentes predefinidas). Devuelve **FALSE** en caso de error.

El formato del archivo de la fuente es binario en la actualidad y dependiente de la arquitectura. Esto quiere decir que usted debería generar los archivos de fuentes en el mismo tipo de CPU que posee la máquina en la que está ejecutando PHP.

Tabla 1. Formato del archivo de fuente

posición de byte	tipo de datos C	descripción
byte 0-3	int	número de caracteres en la fuente
byte 4-7	int	valor del primer caracter en la fuente (con frecuencia 32, indicando el espacio)
byte 8-11	int	ancho de píxel de cada caracter
byte 12-15	int	altura de píxel de cada caracter
byte 16-	char	matriz con datos de caracteres, un byte por píxel en cada caracter, para un total de (n_caracteres*ancho*altura) bytes.

Ejemplo 1. Uso de imageloadfont

```
<?php
$im = imagecreate(50, 20);
$negro = imagecolorallocate($im, 0, 0, 0);
$blanco = imagecolorallocate($im, 255, 255, 255);
imagefilledrectangle($im, 0, 0, 49, 19, $blanco);
$fuente = imageloadfont("04b.gdf");
imagestring($im, $fuente, 0, 0, "Hola", $negro);
imagepng($im);
?>
```

Vea también **imagefontwidth()** y **imagefontheight()**.

imagepalettecopy

(PHP 4 >= 4.0.1, PHP 5)

imagepalettecopy -- Copy the palette from one image to another

Description

int **imagepalettecopy** (resource destination, resource source)

imagepalettecopy() copies the palette from the *source* image to the *destination* image.

imagepng

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

imagepng -- Output a PNG image to either the browser or a file

Description

bool **imagepng** (resource image [, string filename])

The **imagepng()** outputs a GD image stream (*image*) in PNG format to standard output (usually the browser) or, if a filename is given by the *filename* it outputs the image to the file.

```
<?php
$im = imagecreatefrompng("test.png");
imagepng($im);
?>
```

See also **imagegif()**, **imagewbmp()**, **imagejpeg()**, **imagetypes()**.

ImagePolygon

(PHP 3, PHP 4 , PHP 5)

ImagePolygon -- Dibuja un polígono

Descripción

int **imagepolygon** (int im, array points, int num_points, int col)

ImagePolygon crea un polígono en la imagen id. points es un array PHP conteniendo los vértices del polígono. de la siguiente forma points[0] = x0, points1 = y0, points[2] = x1, points[3] = y1, etc. num_points es el número total de vértices.

Vea también **imagecreate()**.

ImagePSBBox

(PHP 3>= 3.0.9, PHP 4 , PHP 5)

ImagePSBBox -- Devuelve el borde que rodea un rectángulo de texto usando fuentes PostScript Type1

Descripción

array **imagepsbbox** (string text, int font, int size, int space, int width, float angle)

size representa pixels.

space perimte cambiar el valor por defecto de un espacio en una fuentes. Este valor es añadido al valor normal y puede ser negativo.

tightness permite controlar la cantidad de espacio en blanco entre carácteres. Este valor se añade a la anchura normal del carácter y puede ser negativo.

angle viene dado en grados.

Los parámetros *space* y *tightness* vienen expresados en unidades de espacio de carácteres, donde una unidad es 1/1000 el borde de una M.

Los parámetros *space*, *tightness* y *angle* son opcionales.

El borde es calculado usando la información disponible de las métricas del carácter, y desafortunadamente tiende a diferir ligeramente de los resultados obtenidos de digitalizar el texto. Si el ángulo es de 0 grados, puede esperar que el texto necesite un pixel más en cada dirección.

Esta función devuelve un array conteniuendo los siguientes elementos:

0	coordenada x inferior izquierda
1	coordenada y inferior izquierda
2	coordenada x superior derecha
3	coordenada y superior derecha

Vea también **imagepstext()**.

ImagePSCopyFont

(PHP 3>= 3.0.9, PHP 4 , PHP 5)

ImagePSCopyFont -- hace una copia de una fuente ya cargada para futuras modificaciones

Descripción

int **imagepscopyfont** (int fontindex)

Use esta función si necesita hacer modificaciones en la fuente, por ejemplo expander/condensar, inclinarla o cambiar su vector de codificación de caracteres, pero también necesita mantener la fuente original. Note que la fuente que quiera copiar debe haber sido obtenida usando **imagepsloadfont()**, no una fuente que sea una copia de otra. Aunque puede hacer modificaciones antes de copiarla.

Si usa esta función, *debe* liberar las fuentes obtenidas de esta manera. De otra forma su script se *colgará*.

En el caso de que todo vaya bien, devolverá un índice de fuente válido que puede ser usado para futuos propósitos. De otra forma la función devolverá **`FALSE`** e imprimirá un mensaje indicando que es lo que ha ido mal.

Vea también **imageploadpsfont()**.

ImagePSEncodeFont

(PHP 3>= 3.0.9, PHP 4 , PHP 5)

ImagePSEncodeFont -- Cambia el vector de codificación de caracteres de una fuente

Descripción

int **imagepsencodefont** (string encodingfile)

Carga un vector de codificación de caracteres desde un archivo y cambia el vector de codificación de las fuentes a él. Loads a character encoding vector from from a file and changes the fonts encoding vector to it. En las fuentes PostScript normalmente faltan muchos caracteres por encima de 127, seguramente quiera cambiar esto si emplea u idioma distinto del inglés.El formato exacto de este archivo está descrito en la documentación de T1libs. T1lib viene con dos archivos listos para usar, IsoLatin1.enc y IsoLatin2.enc.

Si se encuentra usando esta función todo el tiempo, una forma mucho mejor de definir la codificación es establecer ps.default_encoding en el archivo de configuración para que apunte al archivo de codificación correcto y todas las fuentes que cargue tendrán de forma automática la codificación correcta.

imagepsextendfont

(PHP 3>= 3.0.9, PHP 4 , PHP 5)

imagepsextendfont -- Extend or condense a font

Description

bool **imagepsextendfont** (int font_index, float extend)

Extend or condense a font (*font_index*), if the value of the *extend* parameter is less than one you will be condensing the font.

> **Nota:** Esta funcion esta disponible solamente si PHP se ha compilado usando la opcion *--with-t1lib[=DIR]*.

ImagePSFreeFont

(PHP 3>= 3.0.9, PHP 4 , PHP 5)

ImagePSFreeFont -- Libera la memoria usada por un fuente PostScript Type 1

Descripción

void **imagepsfreefont** (int fontindex)

Vea también **imagepsloadfont()**.

imagepsloadfont

(PHP 3>= 3.0.9, PHP 4 , PHP 5)

imagepsloadfont -- Cargar una fuente PostScript Tipo 1 desde un archivo

Descripción

int **imagepsloadfont** (string nombre_archivo)

En caso de que todo haya resultado bien, un índice de fuente válido será devuelto y puede ser usado para propósitos posteriores. De otro modo, la función devuelve **FALSE** e imprime un mensaje describiendo la causa del fallo, el cual no puede leer directamente mientras que el tipo de salida sea una imagen.

Ejemplo 1. Ejemplo de imagepsloadfont()

```
<?php
header("Content-type: image/jpeg");
$im = imagecreate(350, 45);
$negro = imagecolorallocate($im, 0, 0, 0);
$blanco = imagecolorallocate($im, 255, 255, 255);
$fuente = imagepsloadfont("bchbi.pfb"); // o ubique sus archivos .pfb en su maquina
imagepstext($im, "Probando... &iexcl;Funcion&oacute;!", $fuente, 32, $blanco, $negro, 3
imagepsfreefont($fuente);
imagejpeg($im, "", 100); // para la mejor calidad...su experiencia puede ser distinta
imagedestroy($im);
?>
```

> **Nota:** Esta funcion esta disponible solamente si PHP se ha compilado usando la opcion *--with-t1lib[=DIR]*.

Vea también **imagepsfreefont()**.

imagepsslantfont

(PHP 3>= 3.0.9, PHP 4 , PHP 5)

imagepsslantfont -- Slant a font

Description

bool **imagepsslantfont** (int font_index, float slant)

Slant a font given by the *font_index* parameter with a slant of the value of the *slant* parameter.

> **Nota:** Esta funcion esta disponible solamente si PHP se ha compilado usando la opcion *--with-t1lib[=DIR]*.

ImagePSText

(PHP 3>= 3.0.9, PHP 4 , PHP 5)

ImagePSText -- Para dibujar una cadena de texto sobre una imagen usando fuentes PostScript

Type1

Descripción

array **imagepstext** (int image, string text, int font, int size, int foreground, int background, int x, int y [, int space [, int tightness [, float angle [, int antialias_steps]]]])

size viene expresado en pixels.

foreground es el color en el cual el texto será pintado. *background* es el color en el que el texto tratará de resaltar con antialiaing. Los pixels con el color *background* no se pintan, de forma que la imagen de fondo no necesita ser de un color sólido.

Las coordenadas dadas por *x*, *y* definirán el origen (o punto de referencia) del primer carácter (la esquina superior izquierda del carácter). Esto es diferente de la función **ImageString()**, donde x, y definen la esquina superiro derecha del primer carácter. Consulte la documentación de PostScript sobre fuentes y su sistema de medidas si tiene algún problema entendiendo como funcion.

space permite cambiar el valor por defecto de un espacio en la fuente. Esta cantidad es sumada al valor normal y puede ser negativa.

tightness permite controlar la cantidad de espacio en blanco entre caracteres. Esta cantidad es sumada al valor normal y puede ser negativa.

angle viene en grados.

antialias_steps permite controlar el número de colores usados para el antialising del texto. Los valores permitidos son 4 y 16. El valor superior está recomendado para textos con tamaños inferiores a 20, donde el efecto en la calidad del texto es bastante visible. Con tamaños superiores use 4. Hace un menor uso de cálculo.

Parameters *space* y *tightness* están expresados en unidades de espacio de caracteres, donde 1 unidad es 1/1000 de una M mayúscula.

Los parámetros *space*, *tightness*, *angle* y *antialias* son opcionales.

Esta función devuelve una matriz conteniendo los siguientes elementos:

0	coordenada x inferior izquierda
1	coordenada y inferior izquierda
2	coordenada x superior derecha
3	coordenada y superior derecha

Vea también **imagepsbbox()**.

ImageRectangle

(PHP 3, PHP 4 , PHP 5)

ImageRectangle -- Dibuja un rectángulo

Descripción

int **imagerectangle** (int im, int x1, int y1, int x2, int y2, int col)

ImageRectangle crea un rectángulo de color col en la imagen im comenzando en la coordenada superior izquierda x1,y1 y finalizando en la coordenada inferior derecha x2,y2. 0,0 es la esquina superior izquierda de la imagen.

imagerotate

(PHP 4 >= 4.3.0, PHP 5)

imagerotate -- Rotate an image with a given angle

Description

resource **imagerotate** (resource src_im, float angle, int bgd_color)

Rotates the *src_im* image using a given *angle* in degrees. *bgd_color* specifies the color of the uncovered zone after the rotation.

The center of rotation is the center of the image, and the rotated image is scaled down so that the whole rotated image fits in the destination image - the edges are not clipped.

Ejemplo 1. Rotate an image 180 degrees

This example rotates an image 180 degrees - upside down.

```
// File and rotation
$filename = 'test.jpg';
$degrees = 180;

// Content type
header('Content-type: image/jpeg');

// Load
$source = imagecreatefromjpeg($filename);

// Rotate
$rotate = imagerotate($source, $degrees, 0);

// Output
imagejpeg($rotate);
```

> **Nota:** Esta funcion esta disponible solamente si PHP se ha compilado con la version de las bibliotecas GD distribuidas con PHP.

imagesavealpha

(PHP 4 >= 4.3.2, PHP 5)

imagesavealpha -- Set the flag to save full alpha channel information (as opposed to single-color transparency) when saving PNG images

Description

bool **imagesavealpha** (resource image, bool saveflag)

imagesavealpha() sets the flag to attempt to save full alpha channel information (as opposed to single-color transparency) when saving PNG images.

You have to unset alphablending (*imagealphablending($im, **FALSE**)*), to use it.

Alpha channel is not supported by all browsers, if you have problem with your browser, try to load your script with an alpha channel compliant browser, e.g. latest Mozilla.

> **Nota:** Esta funcion requiere GD 2.0.1 o posterior.

See also **imagealphablending()**.

imagesetbrush

(PHP 4 >= 4.0.6, PHP 5)

imagesetbrush -- Set the brush image for line drawing

Description

int **imagesetbrush** (resource image, resource brush)

imagesetbrush() sets the brush image to be used by all line drawing functions (such as **imageline()** and **imagepolygon()**) when drawing with the special colors *IMG_COLOR_BRUSHED* or *IMG_COLOR_STYLEDBRUSHED*.

> **Nota:** You need not take special action when you are finished with a brush, but if you destroy the brush image, you must not use the *IMG_COLOR_BRUSHED* or *IMG_COLOR_STYLEDBRUSHED* colors until you have set a new brush image!

> **Nota:** This function was added in PHP 4.0.6

ImageSetPixel

(PHP 3, PHP 4 , PHP 5)

ImageSetPixel -- Dibuja un pixel

Descripción

int **imagesetpixel** (int im, int x, int y, int col)

ImageSetPixel dibuja un pixel x,y (arriba izquierda 0,0) en la imagen im con color col.

Vea también **imagecreate()** y **imagecolorallocate()**.

imagesetstyle

(PHP 4 >= 4.0.6, PHP 5)

imagesetstyle -- Set the style for line drawing

Description

bool **imagesetstyle** (resource image, array style)

imagesetstyle() sets the style to be used by all line drawing functions (such as **imageline()** and **imagepolygon()**) when drawing with the special color *IMG_COLOR_STYLED* or lines of images with color *IMG_COLOR_STYLEDBRUSHED*. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

The *style* parameter is an array of pixels. Following example script draws a dashed line from upper left to lower right corner of the canvas:

Ejemplo 1. imagesetstyle() example

```
<?php
header("Content-type: image/jpeg");
$im  = imagecreate(100, 100);
$w   = imagecolorallocate($im, 255, 255, 255);
$red = imagecolorallocate($im, 255, 0, 0);

/* Draw a dashed line, 5 red pixels, 5 white pixels */
$style = array($red, $red, $red, $red, $red, $w, $w, $w, $w, $w);
imagesetstyle($im, $style);
imageline($im, 0, 0, 100, 100, IMG_COLOR_STYLED);

/* Draw a line of happy faces using imagesetbrush() with imagesetstyle */
$style = array($w, $w, $w, $w, $w, $w, $w, $w, $w, $w, $w, $w, $red);
imagesetstyle($im, $style);

$brush = imagecreatefrompng("http://www.libpng.org/pub/png/images/smile.happy.png");
$w2 = imagecolorallocate($brush, 255, 255, 255);
imagecolortransparent($brush, $w2);
imagesetbrush($im, $brush);
imageline($im, 100, 0, 0, 100, IMG_COLOR_STYLEDBRUSHED);

imagejpeg($im);
imagedestroy($im);
?>
```

See also **imagesetbrush()**, **imageline()**.

> **Nota:** This function was added in PHP 4.0.6

imagesetthickness

(PHP 4 >= 4.0.6, PHP 5)

imagesetthickness -- Set the thickness for line drawing

Description

bool **imagesetthickness** (resource image, int thickness)

imagesetthickness() sets the thickness of the lines drawn when drawing rectangles, polygons, ellipses etc. etc. to *thickness* pixels. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

> **Nota:** Esta funcion requiere GD 2.0.1 o posterior.

imagesettile

(PHP 4 >= 4.0.6, PHP 5)

imagesettile -- Set the tile image for filling

Description

int **imagesettile** (resource image, resource tile)

imagesettile() sets the tile image to be used by all region filling functions (such as **imagefill()** and **imagefilledpolygon()**) when filling with the special color *IMG_COLOR_TILED*.

A tile is an image used to fill an area with a repeated pattern. *Any* GD image can be used as a tile, and by setting the transparent color index of the tile image with **imagecolortransparent()**, a tile allows certain parts of the underlying area to shine through can be created.

> **Nota:** You need not take special action when you are finished with a tile, but if you destroy the tile image, you must not use the *IMG_COLOR_TILED* color until you have set a new tile image!

ImageString

(PHP 3, PHP 4 , PHP 5)

ImageString -- Dibuja una cadena de texto horizintalmente

Descripción

int **imagestring** (int im, int font, int x, int y, string s, int col)

ImageString dibuja la cadena s en la imagen identificada por im en las coordenadas x,y (arriba izquierda es 0,0) en el color col. Si la fuente es 1, 2, 3, 4 o 5, se emplea una fuente interna.

Vea también **imageloadfont()**.

ImageStringUp

(PHP 3, PHP 4 , PHP 5)

ImageStringUp -- Dibuja una cadena de texto verticalmente

Descripción

int **imagestringup** (int im, int font, int x, int y, string s, int col)

ImageStringUp dibuja la cadena s verticalmente en la imagen identificada por im en las coordenadas x,y (arriba izquierda es 0,0) en el color col. Si la fuente es 1, 2, 3, 4 o 5, se usa una fuente interna.

Vea también **imageloadfont()**.

ImageSX

(PHP 3, PHP 4 , PHP 5)

ImageSX -- Obtiene la anchura de la imagen

Descripción

int **imagesx** (int im)

ImageSX devuelva la anchura de la imagen identificado por im.

Vea también **imagecreate()** y **imagesy()**.

ImageSY

(PHP 3, PHP 4 , PHP 5)

ImageSY -- Obtiene la altura de la imagen

Descripción

int **imagesy** (int im)

ImageSY devuelve la altura de la imagen identificada por im.

Vea también **imagecreate()** y **imagesx()**.

imagetruecolortopalette

(PHP 4 >= 4.0.6, PHP 5)

imagetruecolortopalette -- Convert a true color image to a palette image

Description

void **imagetruecolortopalette** (resource image, bool dither, int ncolors)

imagetruecolortopalette() converts a truecolor image to a palette image. The code for this function was originally drawn from the Independent JPEG Group library code, which is excellent. The code

has been modified to preserve as much alpha channel information as possible in the resulting palette, in addition to preserving colors as well as possible. This does not work as well as might be hoped. It is usually best to simply produce a truecolor output image instead, which guarantees the highest output quality.

dither indicates if the image should be dithered - if it is **TRUE** then dithering will be used which will result in a more speckled image but with better color approximation.

ncolors sets the maximum number of colors that should be retained in the palette.

Nota: Esta funcion requiere GD 2.0.1 o posterior.

imagettfbbox

(PHP 3>= 3.0.1, PHP 4 , PHP 5)

imagettfbbox -- Entrega la caja circundante de un texto usando fuentes TrueType

Descripción

array **imagettfbbox** (float tamanyo, float angulo, string archivo_fuente, string texto)

Esta función calcula y devuelve la caja circundante en pixeles de un texto TrueType.

texto

La cadena a ser medida.

tamanyo

El tamaño de fuente en pixeles.

archivo_fuente

El nombre del archivo de fuente TrueType. (Puede ser también una URL.) Dependiendo de la versión de la biblioteca GD que PHP usa, puede que intente buscar por archivos que no comiencen con un caracter '/', agregando '.ttf' al nombre de archivo y buscando a través de una ruta de fuentes definida por la biblioteca.

angulo

Ã*f*Â ngulo en grados en los que *texto* será medido.

imagettfbbox() devuelve una matriz con 8 elementos que representan cuatro puntos, formando la caja que rodea la caja del texto:

0	esquina inferior izquierda, posición X
1	esquina inferior izquierda, posición Y
2	esquina inferior derecha, posición X

3	esquina inferior derecha, posición Y
4	esquina superior derecha, posición X
5	esquina superior derecha, posición Y
6	esquina superior izquierda, posición X
7	esquina superior izquierda, posición Y

Los puntos son relativos al *texto* independientemente del ángulo, así que "superior izquierdo" quiere decir en la esquina superior a mano izquierda viendo el texto horizontalmente.

Esta función requiere tanto de la biblioteca GD como de la biblioteca FreeType.

Vea también **imagettftext()**.

imagettftext

(PHP 3, PHP 4 , PHP 5)

imagettftext -- Escribir un texto sobre la imagen usando fuentes TrueType

Descripción

array **imagettftext** (resource imagen, float tamanyo, float angulo, int x, int y, int color, string archivo_fuente, string texto)

imagettftext() dibuja la cadena *texto* en la imagen identificada por *imagen*, comenzando en las coordenadas *x*, *y* (la esquina superior izquierda es 0, 0), a un ángulo de *angulo* en el color *color*, usando el archivo de fuente TrueType identificado por *archivo_fuente*. Dependiendo en la versión de la biblioteca GD que PHP usa, cuando *archivo_fuente* no comienza con un caracter '/', '.ttf' se agregará al nombre de archivo y la biblioteca intentará buscar por ese archivo en una ruta de fuentes definida por la biblioteca.

Las coordenadas dadas por *x*, *y* definirán el punto base del primer caracter (a grandes rasgos la esquina inferior izquierda del caracter). Esto a diferencia de **imagestring()**, en donde x, y definen la esquina superior izquierda del primer caracter.

angulo se encuentra en grados, en donde el grado 0 produce la lectura del texto de izquierda a derecha (en dirección de las 3 en punto), y los valores más altos representan una rotación en sentido contrario al de las manecillas del reloj. (Es decir, un valor de 90 resultarío en un texto que se lee de abajo a arriba).

archivo_fuente es la ruta a la fuente TrueType que desea usar.

texto es la cadena de texto, que puede incluir secuencias de caracteres UTF-8 (de la forma: {) para acceder a caracteres más alla de los primeros 255 en una fuente.

color es el índice de color. El uso del valor negativo de un índice de color tiene el efecto de deshabilitar el anti-alias.

imagettftext() devuelve una matriz con 8 elementos que representan cuatro puntos que forman la caja circundante del texto. El orden de los puntos es inferior izquierdo, inferior derecho, superior

derecho y superior izquierdo. Los puntos son relativos al texto independientemente del ángulo, así que "superior izquierdo" quiere decir la esquina del lado superior izquierdo cuando ve el texto horizontalmente.

Este script de ejemplo producirá un JPEG negro de 400x30 pixeles, con las palabras "Probando..." en blanco en la fuente Arial.

Ejemplo 1. Ejemplo de imagettftext()

```
<?php
  header("Content-type: image/jpeg");
  $im = imagecreate(400, 30);
  $blanco = imagecolorallocate($im, 255, 255, 255);
  $negro = imagecolorallocate($im, 0, 0, 0);

  // Reemplaze la ruto con su propio ruta a la fuente
  imagettftext($im, 20, 0, 10, 20, $negro, "/ruta/a/arial.ttf",
  "Probando... Omega: &amp;#937;");
  imagejpeg($im);
  imagedestroy($im);
?>
```

Esta función requiere tanto la biblioteca GD como la biblioteca FreeType.

Vea también **imagettfbbox()**.

imagetypes

(PHP 3 CVS only, PHP 4 >= 4.0.2, PHP 5)

imagetypes -- Return the image types supported by this PHP build

Description

int **imagetypes** (void)

This function returns a bit-field corresponding to the image formats supported by the version of GD linked into PHP. The following bits are returned, **IMG_GIF** | **IMG_JPG** | **IMG_PNG** | **IMG_WBMP** | **IMG_XPM**. To check for PNG support, for example, do this:

Ejemplo 1. imagetypes() example

```
<?php
if (imagetypes() & IMG_PNG) {
    echo "PNG Support is enabled";
}
?>
```

imagewbmp

(PHP 3>= 3.0.15, PHP 4 >= 4.0.1, PHP 5)

imagewbmp -- Output image to browser or file

Description

bool **imagewbmp** (resource image [, string filename [, int foreground]])

imagewbmp() creates the WBMP file in filename from the image *image*. The *image* argument is

the return from the **imagecreate()** function.

The filename argument is optional, and if left off, the raw image stream will be output directly. By sending an image/vnd.wap.wbmp content-type using **header()**, you can create a PHP script that outputs WBMP images directly.

> **Nota:** WBMP support is only available if PHP was compiled against GD-1.8 or later.

Using the optional *foreground* parameter, you can set the foreground color. Use an identifier obtained from **imagecolorallocate()**. The default foreground color is black.

See also **image2wbmp()**, **imagepng()**, **imagegif()**, **imagejpeg()**, **imagetypes()**.

imagexbm

(PHP 5)

imagexbm -- Output XBM image to browser or file

Description

bool **imagexbm** (resource image, string filename [, int foreground])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

> **Nota:** Esta funcion esta disponible solamente si PHP se ha compilado con la version de las bibliotecas GD distribuidas con PHP.

iptcembed

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

iptcembed -- Embed binary IPTC data into a JPEG image

Description

array **iptcembed** (string iptcdata, string jpeg_file_name [, int spool])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

iptcparse

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

iptcparse -- Parse a binary IPTC http://www.iptc.org/ block into single tags.

Description

array **iptcparse** (string iptcblock)

This function parses a binary IPTC block into its single tags. It returns an array using the tagmarker as an index and the value as the value. It returns **FALSE** on error or if no IPTC data was found. See **getimagesize()** for a sample.

jpeg2wbmp

(PHP 4 >= 4.0.5, PHP 5)

jpeg2wbmp -- Convert JPEG image file to WBMP image file

Description

int **jpeg2wbmp** (string jpegname, string wbmpname, int d_height, int d_width, int threshold)

Converts the *jpegname* JPEG file to WBMP format, and saves it as *wbmpname*. With the *d_height* and *d_width* you specify the height and width of the destination image.

> **Nota:** JPEG support is only available if PHP was compiled against GD-1.8 or later.

> **Nota:** WBMP support is only available if PHP was compiled against GD-1.8 or later.

See also **png2wbmp()**.

png2wbmp

(PHP 4 >= 4.0.5, PHP 5)

png2wbmp -- Convert PNG image file to WBMP image file

Description

int **png2wbmp** (string pngname, string wbmpname, int d_height, int d_width, int threshold)

Converts the *pngname* PNG file to WBMP format, and saves it as *wbmpname*. With the *d_height* and *d_width* you specify the height and width of the destination image.

> **Nota:** WBMP support is only available if PHP was compiled against GD-1.8 or later.

See also **jpeg2wbmp()**.

LIII. Funciones IMAP, POP3 y NNTP

Introducción

Estas funciones no se encuentran limitadas al protocolo IMAP, a pesar de su nombre. La biblioteca

c-client interna también soporta métodos de acceso a NNTP, POP3 y buzones de correo locales.

Requirimientos

Esta extensión requiere que la biblioteca c-client se encuentre instalada. Obtenga la versión más reciente de ftp://ftp.cac.washington.edu/imap/ y compílela.

Es importante que no copie los archivos fuente IMAP directamente al directorio de inclusiones del sistema ya que puede crear conflictos. En su lugar, cree un nuevo directorio al interior del directorio de inclusiones del sistema, tal como `/usr/local/imap-2000b/` (la ubicación y nombre dependen de su configuración y versión de IMAP), y al interior de este directorio nuevo cree los directorios adicionales con nombres `lib/` e `include/`. Desde el directorio `c-client` de su árbol de fuentes IMAP, copie todos los archivos `*.h` en `include/` y todos los `*.c` en `lib/`. Adicionalmente, cuando haya compilado IMAP, un archivo llamado `c-client.a` es creado. Coloque también este archivo en el directorio `lib/`, pero cambie su nombre a `libc-client.a`.

> **Nota:** Para compilar la biblioteca c-client con soporte SSL o Kerberos, lea la documentación que viene con el paquete.

Instalación

To get these functions to work, you have to compile PHP with *--with-imap[=DIR]*, where DIR is the c-client install prefix. From our example above, you would use *--with-imap=/usr/local/imap-2000b*. This location depends on where you created this directory according to the description above. Windows users may include the `php_imap.dll` DLL in `php.ini`. IMAP is not supported on systems earlier that Windows 2000. This is because it uses encryption functions in order to enable SSL connections to the mail servers.

> **Nota:** Depending how the c-client was configured, you might also need to add *--with-imap-ssl=/path/to/openssl/* and/or *--with-kerberos=/path/to/kerberos* into the PHP configure line.

Aviso
La extensión IMAP no puede ser usada junto con las extensiones recode, YAZ ó Cyrus. Esto es debido a que las dos utilizan el mismo símbolo interno

---

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

NIL (integer)

OP_DEBUG (integer)

OP_READONLY (integer)

Open mailbox read-only

OP_ANONYMOUS (integer)

Don't use or update a `.newsrc` for news (NNTP only)

OP_SHORTCACHE (integer)

OP_SILENT (integer)

OP_PROTOTYPE (integer)

OP_HALFOPEN (integer)

For IMAP and NNTP names, open a connection but don't open a mailbox.

OP_EXPUNGE (integer)

OP_SECURE (integer)

CL_EXPUNGE (integer)

silently expunge the mailbox before closing when calling **imap_close()**

FT_UID (integer)

The parameter is a UID

FT_PEEK (integer)

Do not set the \Seen flag if not already set

FT_NOT (integer)

FT_INTERNAL (integer)

The return string is in internal format, will not canonicalize to CRLF.

FT_PREFETCHTEXT (integer)

ST_UID (integer)

The sequence argument contains UIDs instead of sequence numbers

ST_SILENT (integer)

ST_SET (integer)

CP_UID (integer)

the sequence numbers contain UIDS

CP_MOVE (integer)

Delete the messages from the current mailbox after copying with **imap_mail_copy()**

SE_UID (integer)

Return UIDs instead of sequence numbers

SE_FREE (integer)

SE_NOPREFETCH (integer)

Don't prefetch searched messages

SO_FREE (integer)

SO_NOSERVER (integer)

SA_MESSAGES (integer)

SA_RECENT (integer)

SA_UNSEEN (integer)

SA_UIDNEXT (integer)

SA_UIDVALIDITY (integer)

SA_ALL (integer)

LATT_NOINFERIORS (integer)

This mailbox has no "children" (there are no mailboxes below this one).

LATT_NOSELECT (integer)

This is only a container, not a mailbox - you cannot open it.

LATT_MARKED (integer)

This mailbox is marked. Only used by UW-IMAPD.

LATT_UNMARKED (integer)

This mailbox is not marked. Only used by UW-IMAPD.

SORTDATE (integer)

Sort criteria for **imap_sort()**: message Date

SORTARRIVAL (integer)

Sort criteria for **imap_sort()**: arrival date

SORTFROM (integer)

Sort criteria for **imap_sort()**: mailbox in first From address

SORTSUBJECT (integer)

Sort criteria for **imap_sort()**: message subject

SORTTO (integer)

Sort criteria for **imap_sort()**: mailbox in first To address

SORTCC (integer)

Sort criteria for **imap_sort()**: mailbox in first cc address

SORTSIZE (integer)

Sort criteria for **imap_sort()**: size of message in octets

TYPETEXT (integer)

TYPEMULTIPART (integer)

TYPEMESSAGE (integer)

TYPEAPPLICATION (integer)

TYPEAUDIO (integer)

TYPEIMAGE (integer)

TYPEVIDEO (integer)

TYPEOTHER (integer)

ENC7BIT (integer)

ENC8BIT (integer)

ENCBINARY (**integer**)

ENCBASE64 (**integer**)

ENCQUOTEDPRINTABLE (**integer**)

ENCOTHER (**integer**)

Ver también

Este documento no puede entrar en detalles sobre todos los temas que involucran lan funciones ofrecidas. Puede encontrar más información en la documentación de las fuentes de la biblioteca c-client (`docs/internal.txt`), y en los siguientes documentos RFC:

- RFC2821: Protocolo Simple de Transferencia de Correo (SMTP).
- RFC2822: Estándar para los mensajes de texto en internet ARPA.
- RFC2060: Protocolo de Acceso de Mensaje de Internet (IMAP) Versión 4rev1.
- RFC1939: Protocolo Post Office Versión 3 (POP3).
- RFC977: Protocolo de Transferencia de Noticias en Red (NNTP).
- RFC2076: Cabeceras de Mensajes de Internet Comunes.
- RFC2045 , RFC2046 , RFC2047 , RFC2048 & RFC2049: Extensiones de Correo de Internet Multi-propósito (MIME).

Una vista general detallada se encuentra disponible también en los libros Programming Internet Email por David Wood y Managing IMAP por Dianna Mullet y Kevin Mullet.

Tabla de contenidos

imap_fetchheader -- Devuelve la cabecera de un mensaje
imap_fetchstructure -- Lee la estructuta de un mensaje concreto
imap_get_quota -- Retrieve the quota level settings, and usage statics per mailbox
imap_get_quotaroot -- Retrieve the quota settings per user
imap_getacl -- Gets the ACL for a given mailbox
imap_getmailboxes -- Lee la lista de buzones, devolviendo informacón detallada de cada uno
imap_getsubscribed -- Lista todos los buzones subscritos
imap_header -- Lee la cabecera del mensaje
imap_headerinfo -- Read the header of the message
imap_headers -- Returns headers for all messages in a mailbox
imap_last_error -- Esta función devuelve el último error IMAP (si se produjo) que ocurrió durante la petición de esta página.
imap_list -- Read the list of mailboxes
imap_listmailbox -- Lee la lista de buzones
imap_listscan -- Read the list of mailboxes, takes a string to search for in the text of the mailbox
imap_listsubscribed -- Lista todos los buzones subscritos
imap_lsub -- List all the subscribed mailboxes
imap_mail_compose -- Create a MIME message based on given envelope and body sections
imap_mail_copy -- Copia los mensajes especificados a un buzón
imap_mail_move -- Mueve los mensajes especificados a un buzón
imap_mail -- Send an email message
imap_mailboxmsginfo -- Obtiene información acerca del buzón actual
imap_mime_header_decode -- Decode MIME header elements
imap_msgno -- Esta función devuelve el número de secuencia del mensaje para el UID dado.
imap_num_msg -- Informa del número de mensajes en el buzón actual
imap_num_recent -- Informa el número de mensajes recientes en el buzón actual
imap_open -- Abre una sesión IMAP
imap_ping -- Comprueba si la sesión IMAP está aún activa
imap_qprint -- Convierte una cadena quoted-printable a una cadena de 8 bit
imap_renamemailbox -- Renombra un buzón
imap_reopen -- Reabre una sesión IMAP a un nuevo buzón
imap_rfc822_parse_adrlist -- Examina la cadena dirección
imap_rfc822_parse_headers -- Parse mail headers from a string
imap_rfc822_write_address -- Devuleve una dirección de correo correctamente formateada dado el buzón, host, e información personal.
imap_scanmailbox -- Lee la lista de buzones y toma una cadena para buscar en el texto del buzón
imap_search -- Esta función devuelve un array de mensajes que coinciden con el criterio de busqueda dado.
imap_set_quota -- Sets a quota for a given mailbox
imap_setacl -- Sets the ACL for a giving mailbox
imap_setflag_full -- Activa flags en los mensajes
imap_sort -- Ordena un array de cabeceras de mensajes
imap_status -- Esta función devuelve el información de estado de otro buzón distinto al actual.
imap_subscribe -- Subscribe to a mailbox
imap_thread -- Returns a tree of threaded message
imap_timeout -- Set or fetch imap timeout
imap_uid -- Esta función devuelve el UID del número de secuencia del mensaje dado
imap_undelete -- Desmarca los mensajes que están marcados como borrados
imap_unsubscribe -- Unsubscribe from a mailbox
imap_utf7_decode -- Decodes a modified UTF-7 encoded string
imap_utf7_encode -- Converts ISO-8859-1 string to modified UTF-7 text
imap_utf8 -- Converts MIME-encoded text to UTF-8

imap_8bit

(PHP 3, PHP 4 , PHP 5)

imap_8bit -- Convierte una cadena de 8bit a una cadena quoted-printable

Descripción

string **imap_8bit** (string string)

Convierte una cadena de 8bit a una cadena quoted-printable.

Devuelve una cadena quoted-printable

imap_alerts

(PHP 3>= 3.0.12, PHP 4 , PHP 5)

imap_alerts -- Esta función devuelve todos los mensajes de alerta IMAP (si hubo) que han ocurrido durante la petición de la pagina o desde que la pila de alertas fue inicializada.

Descripción

array **imap_alerts** (void)

Esta función devuelve un array con todos los mensajes de alerta IMAP generados desde la última llamada a **imap_alerts()**, o el comienzo de la pagina. Cuando se llama a **imap_alerts()**, la pila de alerta es inicializada. La especificación IMAP requiere que estos mensajes sean pasados al usuario.

imap_append

(PHP 3, PHP 4 , PHP 5)

imap_append -- Agrega una cadena de mensaje al buzón especificado

Descripción

int **imap_append** (int imap_stream, string mbox, string message, string flags)

Devuelve **`TRUE`** si no hay error y **`FALSE`** en caso contrario.

imap_append() agrega una cadena de mensaje al buzón especificado *mbox*. Si se especifica el parámetro *flags*, escribe las opciones o condiciones establecidas en el parámetro *flags* al buzón.

Cuando conecte con el servidor Cyrus IMAP, debe usar "\r\n" como finalizador de linea en vez de "\n" o la operación fallará.

imap_base64

(PHP 3, PHP 4 , PHP 5)

imap_base64 -- Decodifica texto codificado en BASE64

Descripción

string **imap_base64** (string text)

imap_base64() decodifica texto codificado en BASE-64. El mensaje decodificado es devuelto como una cadena.

imap_binary

(PHP 3>= 3.0.2, PHP 4 , PHP 5)

imap_binary -- Convierte una cadena de 8bit a una cadena base64

Descripción

string **imap_binary** (string string)

Convierte una cadena de 8bit a una cadena base64.

Devuleve una cadena base64.

imap_body

(PHP 3, PHP 4 , PHP 5)

imap_body -- Lee el cuerpo del mensaje

Descripción

string **imap_body** (int imap_stream, int msg_number, int flags)

imap_body() devuelve el cuerpo del mensaje, numerado *msg_number* del buzón actual. Los *flags* opcionales son una máscara de bit con una o mas de las siguientes:

- FT_UID - El msgno es un UID
- FT_PEEK - No activar \Seen flag si no está ya activa
- FT_INTERNAL - La cadena devuelta está en formato interno, no canoniza a CRLF.

imap_bodystruct

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

imap_bodystruct -- Read the structure of a specified body section of a specific message

Description

object **imap_bodystruct** (resource stream_id, int msg_no, string section)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

imap_check

(PHP 3, PHP 4 , PHP 5)

imap_check -- Comprueba el estado del buzón actual

Descripción

object **imap_check** (int imap_stream)

Devuelve información acerca del buzón actual. Devuelve **`FALSE`** si falla.

La función **imap_check()** comprueba el estado del buzón actual en el servidor y devuelve la información en un objeto con las siguientes propiedades.

Date : fecha del mensaje
Driver : controlador
Mailbox : nombre del buzón
Nmsgs : número de mensajes
Recent : número de mensajes recientes

imap_clearflag_full

(PHP 3>= 3.0.3, PHP 4 , PHP 5)

imap_clearflag_full -- Limpia los flags de los mensajes

Descripción

string **imap_clearflag_full** (int stream, string sequence, string flag, string options)

Esta funcion elimina el flag especificado del conjunto de flags activos para los mensajes en la secuencia especificada.

Las opciones son una máscara de bit con uno o más de los siguientes:

ST_UID El argumento sequence contiene UIDs en vez de

números secuenciales

imap_close

(PHP 3, PHP 4 , PHP 5)

imap_close -- Cierra una sesión IMAP

Descripción

int **imap_close** (int imap_stream, int flags)

Cierra una sesión imap. Toma un parámetro *flag* opcional, CL_EXPUNGE, el cual purgará el buzón de forma trasparente antes de cerrarla.

imap_createmailbox

(PHP 3, PHP 4 , PHP 5)

imap_createmailbox -- Crea un buzón nuevo

Descripción

int **imap_createmailbox** (int imap_stream, string mbox)

imap_createmailbox() crea un buzón nuevo especificado por *mbox* (ver **imap_open()** para el formato del parámetro *mbox*).

Devuelve **`TRUE`** si no hay error y **`FALSE`** en caso contrario.

Ver También **imap_renamemailbox()** y **imap_deletemailbox()**.

imap_delete

(PHP 3, PHP 4 , PHP 5)

imap_delete -- Marca un mensaje para ser borrado en el buzón actual

Descripción

int **imap_delete** (int imap_stream, int msg_number)

Devuelve **`TRUE`**.

La función **imap_delete()** marca el mensaje referenciado por *msg_number* para su eliminación. El borrado físico de los mensajes es realizado por **imap_expunge()**.

imap_deletemailbox

(PHP 3, PHP 4 , PHP 5)

imap_deletemailbox -- Elimina un buzón

Descripción

int **imap_deletemailbox** (int imap_stream, string mbox)

imap_deletemailbox() elimina el buzón especificado (ver **imap_open()** para el formato del *mbox*).

Devuelve **TRUE** si no hay error y **FALSE** en caso contrario.

Ver También **imap_createmailbox()** y **imap_reanmemailbox()**.

imap_errors

(PHP 3>= 3.0.12, PHP 4 , PHP 5)

imap_errors -- Esta función devuelve todos los errores IMAP (si hubo) que han occurrido durante la petición de la página o desde que la pila de errores se inicializó.

Descripción

array **imap_errors** (void)

Esta función devuelve un array de todos los mensajes de error IMAP generados desde la última llamada a **imap_errors()**, o el principio de la página. Cuando se llama a **imap_errors()**, la pila de errores se inicializa.

ATENCIÃƒ N: esta función no esta disponible aún en PHP4.

imap_expunge

(PHP 3, PHP 4 , PHP 5)

imap_expunge -- Elimina todos los mensajes marcados como borrados

Descripción

int **imap_expunge** (int imap_stream)

imap_expunge() elimina todos los mensajes marcados por la función **imap_delete()**.

Devuleve **TRUE**.

imap_fetch_overview

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

imap_fetch_overview -- Read an overview of the information in the headers of the given message

Description

array **imap_fetch_overview** (resource imap_stream, string sequence [, int options])

This function fetches mail headers for the given *sequence* and returns an overview of their contents. *sequence* will contain a sequence of message indices or UIDs, if *flags* contains FT_UID. The returned value is an array of objects describing one message header each:

- subject - the messages subject
- from - who sent it
- date - when was it sent
- message_id - Message-ID
- references - is a reference to this message id
- size - size in bytes
- uid - UID the message has in the mailbox
- msgno - message sequence number in the mailbox
- recent - this message is flagged as recent
- flagged - this message is flagged
- answered - this message is flagged as answered
- deleted - this message is flagged for deletion
- seen - this message is flagged as already read
- draft - this message is flagged as being a draft

Ejemplo 1. imap_fetch_overview() example

```
<?php
$mbox = imap_open("{your.imap.host:143}", "username", "password")
     or die("can't connect: " . imap_last_error());

$overview = imap_fetch_overview($mbox, "2,4:6", 0);

if (is_array($overview)) {
        reset($overview);
        while (list($key, $val) = each($overview)) {
                echo      $val->msgno
                . " - " . $val->date
                . " - " . $val->subject
                . "\n";
        }
}

imap_close($mbox);
?>
```

imap_fetchbody

(PHP 3, PHP 4 , PHP 5)

imap_fetchbody -- Localiza una sección particular en el cuerpo del mensaje

Descripción

string **imap_fetchbody** (int imap_stream, int msg_number, string part_number, flags flags)

Esta función busca una sección particular en el cuerpo de los mensajes especificados, como una cadena de texto y devuleve esa cadena. La especificación de la sección es una cadena de enteros delimitados por comas,los cuales indexan las partes del cuerpo como indica la especificación IMAP4. Partes del cuerpo no son decodificadas por esta función.

Las opciones para **imap_fetchbody ()** son una máscara de bit con una o más de las siguientes

- FT_UID - El msgono es un UID
- FT_PEEK - No activar \Seen flag si no esta ya activa
- FT_INTERNAL - La cadena devuelta está en formato "interno", sin ningún intento por canonizar CRLF

imap_fetchheader

(PHP 3>= 3.0.3, PHP 4 , PHP 5)

imap_fetchheader -- Devuelve la cabecera de un mensaje

Descripción

string **imap_fetchheader** (resource secuencia_imap, int num_mensaje [, int opciones])

Esta función produce la recuperación de la cabecera completa, sin filtrar, en formato RFC2822 del mensaje especificado, como una cadena de texto y devuelve esa cadena de texto.

Las *opciones* son:

- **`FT_UID`** - El argumento *num_mensaje* es un UID
- **`FT_INTERNAL`** - La cadena de retorno está en formato "interno", sin intento alguno de moldear a nuevas líneas tipo CRLF
- **`FT_PREFETCHTEXT`** - El archivo RFC822.TEXT debe ser pre-solicitado al mismo tiempo. Esto evita un RTT extra en una conexión IMAP si se desea un mensaje de texto completo (p.ej. en la operación "guardar a archivo local")

imap_fetchstructure

(PHP 3, PHP 4 , PHP 5)

imap_fetchstructure -- Lee la estructuta de un mensaje concreto

Descripción

object **imap_fetchstructure** (int imap_stream, int msg_number [, int flags])

Esta función busca toda la información estructurada en el mensaje especificado. El parámetro opcional *flags* sólo tiene una opcion, *FT_UID*, la cual indica a la función que trate el argumento *msg_number* como un *UID*. El objeto devuelto incluye el sobre, la fecha interna, el tamaÃ‚Â¤o, flags y la estructura del cuerpo con un objeto similar por cada mime adjunto al mensaje. La estructura de los objetos devueltos es como sigue:

Tabla 1. Objetos Devueltos para imap_fetchstructure()

type	Tipo primario del cuerpo
encoding	Body transfer encoding
ifsubtype	**`TRUE`** si hay una cadena de subtipo
subtype	MIME subtype
ifDescripción	**`TRUE`** si hay una cadena de Descripción
Description	Conenido de la cadena de Descripción
ifid	**`TRUE`** si hay una cadena de identificación
id	Cadena de Identificación
lines	Número de lineas
bytes	Número de bytes
ifdisposition	**`TRUE`** si hay una cadena de configuración
disposition	Cadena de configuración
ifdparameters	**`TRUE`** si el array dparameters existe
dparameters [a]	Array de parametro de configuración
ifparameters	**`TRUE`** si el array de parámetros existe
parameters [b]	MIME parameters array

parts [c]	Array de objetos describiendo cada parte del mensaje
Notas de Tabla: a. dparameters es un array de objetos donde cada objeto tiene un "atributo" y una propiedad "valor". b. parameter es un array de objetos donde cada objetor tiene un "atributo" y una propiedad "valor". c. parts es un array de objetos identico en estructura al objeto del primer nivel, con la limitación de que este no puede contener más objetos 'parts'.	

Tabla 2. Tipo primario del cuerpo

0	texto
1	multiparte
2	mensaje
3	aplicación
4	audio
5	imagen
6	video
7	otro

Tabla 3. Codificacion para tranferencia

0	7BIT
1	8BIT
2	BINARY
3	BASE64
4	QUOTED-PRINTABLE
5	OTRO

imap_get_quota

(PHP 4 >= 4.0.5, PHP 5)

imap_get_quota -- Retrieve the quota level settings, and usage statics per mailbox

Description

array **imap_get_quota** (resource imap_stream, string quota_root)

Returns an array with integer values limit and usage for the given mailbox. The value of limit represents the total amount of space allowed for this mailbox. The usage value represents the mailboxes current level of capacity. Will return **FALSE** in the case of failure.

This function is currently only available to users of the c-client2000 or greater library.

NOTE: For this function to work, the mail stream is required to be opened as the mail-admin user. For a non-admin user version of this function, please see the **imap_get_quotaroot()** function of PHP.

imap_stream should be the value returned from an **imap_open()** call. NOTE: This stream is required to be opened as the mail admin user for the get_quota function to work. *quota_root* should normally be in the form of user.name where name is the mailbox you wish to retrieve information about.

Ejemplo 1. imap_get_quota() example

```
<?php
$mbox = imap_open("{your.imap.host}", "mailadmin", "password", OP_HALFOPEN)
      or die("can't connect: " . imap_last_error());

$quota_value = imap_get_quota($mbox, "user.kalowsky");
if (is_array($quota_value)) {
    echo "Usage level is: " . $quota_value['usage'];
    echo "Limit level is: " . $quota_value['limit'];
}

imap_close($mbox);
?>
```

As of PHP 4.3, the function more properly reflects the functionality as dictated by the RFC 2087. The array return value has changed to support an unlimited number of returned resources (i.e. messages, or sub-folders) with each named resource receiving an individual array key. Each key value then contains an another array with the usage and limit values within it. The example below shows the updated returned output.

For backwards compatibility reasons, the original access methods are still available for use, although it is suggested to update.

Ejemplo 2. imap_get_quota() 4.3 or greater example

```
<?php
$mbox = imap_open("{your.imap.host}", "mailadmin", "password", OP_HALFOPEN)
      or die("can't connect: " . imap_last_error());

$quota_values = imap_get_quota($mbox, "user.kalowsky");
if (is_array($quota_values)) {
   $storage = $quota_values['STORAGE'];
   echo "STORAGE usage level is: " .  $storage['usage'];
   echo "STORAGE limit level is: " .  $storage['limit'];

   $message = $quota_values['MESSAGE'];
   echo "MESSAGE usage level is: " .  $message['usage'];
   echo "MESSAGE limit is: " .  $message['limit'];

   /* ...  */
}

imap_close($mbox);
?>
```

See also **imap_open()**, **imap_set_quota()** and **imap_get_quotaroot()**.

imap_get_quotaroot

(PHP 4 >= 4.3.0, PHP 5)

imap_get_quotaroot -- Retrieve the quota settings per user

Description

array **imap_get_quotaroot** (resource imap_stream, string quota_root)

Returns an array of integer values pertaining to the specified user mailbox. All values contain a key based upon the resource name, and a corresponding array with the usage and limit values within.

The limit value represents the total amount of space allowed for this user's total mailbox usage. The usage value represents the user's current total mailbox capacity. This function will return **FALSE** in the case of call failure, and an array of information about the connection upon an un-parsable response from the server.

This function is currently only available to users of the c-client2000 or greater library.

imap_stream should be the value returned from an **imap_open()** call. This stream should be opened as the user whose mailbox you wish to check. *quota_root* should normally be in the form of which mailbox (i.e. INBOX).

Ejemplo 1. imap_get_quotaroot() example

```
<?php
$mbox = imap_open("{your.imap.host}", "kalowsky", "password", OP_HALFOPEN)
      or die("can't connect: " . imap_last_error());

$quota = imap_get_quotaroot($mbox, "INBOX");
if (is_array($quota)) {
   $storage = $quota_values['STORAGE'];
   echo "STORAGE usage level is: " .  $storage['usage'];
   echo "STORAGE limit level is: " .  $storage['limit'];

   $message = $quota_values['MESSAGE'];
   echo "MESSAGE usage level is: " .  $message['usage'];
   echo "MESSAGE usage level is: " .  $message['limit'];

   /* ...  */

}

imap_close($mbox);
?>
```

See also **imap_open()**, **imap_set_quota()** and **imap_get_quota()**.

imap_getacl

(PHP 5)

imap_getacl -- Gets the ACL for a given mailbox

Description

array **imap_getacl** (resource stream_id, string mailbox)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

This function is currently only available to users of the c-client2000 or greater library.

See also **imap_setacl()**.

imap_getmailboxes

(PHP 3>= 3.0.12, PHP 4 , PHP 5)

imap_getmailboxes -- Lee la lista de buzones, devolviendo informacón detallada de cada uno

Descripción

array **imap_getmailboxes** (int imap_stream, string ref, string pat)

Devuelve un array de objetos coneniendo información del buzón. Cada objeto tiene los atributos *name*, especificando el nombre completo del buzón; *delimiter*, que es el delimitador jerárquico para la parte de la jerarquía dónde está este buzón; y *attributes*. *Attributes* es una máscara de bits contra la que se puede probar:

- LATT_NOINFERIORS - Este buzón no tiene "hijos" (No ha buzones por debajo de él)
- LATT_NOSELECT - Esto es sólo un contenedor, no un buzón - No puede abrirlo.
- LATT_MARKED - Este buzón está marcado. Unicamente usado por UW-IMAPD.
- LATT_UNMARKED - Este buzón no está marcado. Unicamente usado por UW-IMAPD.

ref normalmente debería ser solo el servidor IMAP, de la forma: {imap_server:imap_port}, y *pattern* especifica, dónde en la estructura jerárquica del buzón, para comenzar a buscar. Si quiere todo los buzones, pase el parámetro *pattern* como una cadena vacía.

Hay dos caracteres especiales que puede pasar como parte del parámetro *pattern*: '*' and '%'. '*' significa que devuelva todos los buzones. Si pasa el parámetro *pattern* como '*', obtendrá una lista con la jerarquía completa del buzón. %' significa que devuelva sólo el nivel actual. Pasar '%' en el parámetro *pattern* devolverá sólo el nivel más alto de los buzones; '~/mail/%' en UW_IMAPD devolverá cada buzón del directorio ~/mail, pero ninguno de los subdirectorios de ese directorio.

imap_getsubscribed

(PHP 3>= 3.0.12, PHP 4 , PHP 5)

imap_getsubscribed -- Lista todos los buzones subscritos

Descripción

array **imap_getsubscribed** (int imap_stream, string ref, string pattern)

Esta función es idéntica a **imap_getmailboxes()**, excepto que esta sólo devuelve los buzones a los que está subscrito el usuario.

imap_header

(PHP 3, PHP 4 , PHP 5)

imap_header -- Lee la cabecera del mensaje

Descripción

object **imap_header** (int imap_stream, int msg_number [, int fromlength [, int subjectlength [, string defaulthost]]])

Esta función devuelve un objeto con varios elementos de la cabecera.

remail, date, Date, subject, Subject, in_reply_to, message_id,
newsgroups, followup_to, references

message flags:
Recent - 'R' si es reciente y ha sido leido,
'N' si es reciente y no ha sido leido,
' ' si no es reciente
Unseen - 'U' si no ha sido leido Y no es reciente,
' ' si ha sido leido O no y es reciente
Answered -'A' si ha sido contestado,
' ' si no ha sido contestado
Deleted - 'D' si ha sido borrado,
' ' si no ha sido borrado
Draft - 'X' if draft,
' ' if not draft
Flagged - 'F' si esta if flagged,
' ' if not flagged

OBSERVE que el comportamiento Recent/Unseen es un poco extraÃ‚Â¤o. Si quiere conocer si un mensaje es Unseen, debe comprobarlo asi

Unseen == 'U' || Recent == 'N'

toaddress (la linea to: al completo, hasta 1024 caracteres)

to[] (devuelve un array de objetos a partir de la linea To, conteniendo:)
personal
adl
mailbox
host

fromaddress (la linea from: al completo, hasta 1024 caracteres)

from[] (devuelve un array de objetos a partir de la linea From, conteniendo:)
personal
adl
mailbox
host

ccaddress (la linea cc: al completo, hasta 1024 caracteres)
cc[] (devuelve un array de objetos a partir de la linea Cc:, conteniendo:)
  personal
  adl
  mailbox
  host

bccaddress (la linea bcc al completo, hasta 1024 caracteres)
bcc[] (devuelve un array de objetos a partir de la linea Bcc, conteniendo:)
  personal
  adl
  mailbox
  host

reply_toaddress (la linea reply_to: al completo, hasta 1024 caracteres)
reply_to[] (devuelve un array de objetos a partir de la linea Reply_to,
conteniendo:)
  personal
  adl
  mailbox
  host

senderaddress (la linea sender: al completo, hasta 1024 caracteres)
sender[] (devuelve un array de objetos a partir de la linea sender, conteniendo:)
  personal
  adl
  mailbox
  host

return_path (la linea return-path: al completo, hasta 1024 caracteres)
return_path[] (devuelve un array de objetos a partir de la linea return_path,
conteniendo:)
  personal
  adl
  mailbox
  host

udate (fecha del mensaje en formato unix)

fetchfrom (la linea from formateada hasta ajustarse a los caracteres
indicados en *fromlength*)

fetchsubject (la linea subject formateada hasta ajustarse a los caracteres
indicados en *subjectlength*)

imap_headerinfo

(PHP 3, PHP 4 , PHP 5)

imap_headerinfo -- Read the header of the message

Description

object **imap_headerinfo** (resource imap_stream, int msg_number [, int fromlength [, int subjectlength [, string defaulthost]]])

This function returns an object of various header elements.

remail, date, Date, subject, Subject, in_reply_to, message_id,
newsgroups, followup_to, references

message flags:
Recent - 'R' if recent and seen,
'N' if recent and not seen,
' ' if not recent
Unseen - 'U' if not seen AND not recent,
' ' if seen OR not seen and recent
Answered -'A' if answered,
' ' if unanswered
Deleted - 'D' if deleted,
' ' if not deleted
Draft - 'X' if draft,
' ' if not draft
Flagged - 'F' if flagged,
' ' if not flagged

NOTE that the Recent/Unseen behavior is a little odd. If you want to know if a message is Unseen, you must check for

Unseen == 'U' || Recent == 'N'

toaddress (full to: line, up to 1024 characters)

to[] (returns an array of objects from the To line, containing):
personal
adl
mailbox
host

fromaddress (full from: line, up to 1024 characters)

from[] (returns an array of objects from the From line, containing):
personal
adl
mailbox
host

ccaddress (full cc: line, up to 1024 characters)
cc[] (returns an array of objects from the Cc line, containing):
personal
adl
mailbox
host

bccaddress (full bcc line, up to 1024 characters)
bcc[] (returns an array of objects from the Bcc line, containing):
  personal
  adl
  mailbox
  host

reply_toaddress (full reply_to: line, up to 1024 characters)
reply_to[] (returns an array of objects from the Reply_to line,
containing):
  personal
  adl
  mailbox
  host

senderaddress (full sender: line, up to 1024 characters)
sender[] (returns an array of objects from the sender line, containing):
  personal
  adl
  mailbox
  host

return_path (full return-path: line, up to 1024 characters)
return_path[] (returns an array of objects from the return_path line,
containing):
  personal
  adl
  mailbox
  host

udate (mail message date in unix time)

fetchfrom (from line formatted to fit *fromlength*
characters)

fetchsubject (subject line formatted to fit *subjectlength* characters)

imap_headers

(PHP 3, PHP 4 , PHP 5)

imap_headers -- Returns headers for all messages in a mailbox

Descripción

array **imap_headers** (int imap_stream)

Devuelve un array de cadenas formateadas con informacion de la cabecera. Un elemento por mensaje de correo.

imap_last_error

(PHP 3>= 3.0.12, PHP 4 , PHP 5)

imap_last_error -- Esta función devuelve el último error IMAP (si se produjo) que ocurrió durante la petición de esta página.

Descripción

string **imap_last_error** (void)

Esta función devuelve el texto completo del último error IMAP que ocurrió en la pagina actual. La plia de errores The error stack is untouched; llamando despues a la función **imap_last_error()**, sin que se produzca un error, devolverá el mismo error.

ATENCIÃƒ N: esta función no esta disponible aún en PHP4.

imap_list

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

imap_list -- Read the list of mailboxes

Description

array **imap_list** (resource imap_stream, string ref, string pattern)

Returns an array containing the names of the mailboxes. See **imap_getmailboxes()** for a description of *ref* and *pattern*.

Ejemplo 1. imap_list() example

```
<?php
$mbox = imap_open("{your.imap.host}", "username", "password", OP_HALFOPEN)
      or die("can't connect: " . imap_last_error());

$list = imap_list($mbox, "{your.imap.host}", "*");
if (is_array($list)) {
  reset($list);
  while (list($key, $val) = each($list)) {
    echo imap_utf7_decode($val) . "<br />\n";
  }
} else {
  echo "imap_list failed: " . imap_last_error() . "\n";
}

imap_close($mbox);
?>
```

See also: **imap_getmailboxes()**.

imap_listmailbox

(PHP 3, PHP 4 , PHP 5)

imap_listmailbox -- Lee la lista de buzones

Descripción

array **imap_listmailbox** (int imap_stream, string ref, string pat)

Devuelve un array que contiene los nombres de los buzones.

imap_listscan

(no version information, might be only in CVS)

imap_listscan -- Read the list of mailboxes, takes a string to search for in the text of the mailbox

Description

array **imap_listscan** (resource imap_stream, string ref, string pattern, string content)

Returns an array containing the names of the mailboxes that have *content* in the text of the mailbox.

This function is similar to **imap_listmailbox()**, but it will additionally check for the presence of the string *content* inside the mailbox data.

See **imap_getmailboxes()** for a description of *ref* and *pattern*.

imap_listsubscribed

(PHP 3, PHP 4 , PHP 5)

imap_listsubscribed -- Lista todos los buzones subscritos

Descripción

array **imap_listsubscribed** (int imap_stream, string ref, string pattern)

Devuelve un array de todos los buzones que usted tiene subscritos. Los parámetros *ref* y *pattern* especifican la localización desde donde comenzará a buscar y el patrón que el nombre del buzón debe encontrar.

imap_lsub

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

imap_lsub -- List all the subscribed mailboxes

Description

array **imap_lsub** (resource imap_stream, string ref, string pattern)

Returns an array of all the mailboxes that you have subscribed.

imap_mail_compose

(PHP 3>= 3.0.5, PHP 4 , PHP 5)

imap_mail_compose -- Create a MIME message based on given envelope and body sections

Description

string **imap_mail_compose** (array envelope, array body)

Ejemplo 1. imap_mail_compose() example

```
<?php

$envelope["from"]= "joe@example.com";
$envelope["to"]  = "foo@example.com";
$envelope["cc"]  = "bar@example.com";

$part1["type"] = TYPEMULTIPART;
$part1["subtype"] = "mixed";

$filename = "/tmp/imap.c.gz";
$fp = fopen($filename, "r");
$contents = fread($fp, filesize($filename));
fclose($fp);

$part2["type"] = TYPEAPPLICATION;
$part2["encoding"] = ENCBINARY;
$part2["subtype"] = "octet-stream";
$part2["description"] = basename($filename);
$part2["contents.data"] = $contents;

$part3["type"] = TYPETEXT;
$part3["subtype"] = "plain";
$part3["description"] = "description3";
$part3["contents.data"] = "contents.data3\n\n\n\t";

$body[1] = $part1;
$body[2] = $part2;
$body[3] = $part3;

echo nl2br(imap_mail_compose($envelope, $body));

?>
```

imap_mail_copy

(PHP 3, PHP 4 , PHP 5)

imap_mail_copy -- Copia los mensajes especificados a un buzón

Descripción

int **imap_mail_copy** (int imap_stream, string msglist, string mbox, int flags)

Devuelve **TRUE** si no hay error y **FALSE** en caso contrario.

Copia los mensajes especificados por *msglist* a un buzón especificado. *msglist* es un rango no números de mensajes.

Flags es una máscara de bit de uno o más

- CP_UID - los números de secuencia contienen UIDS
- CP_MOVE - Elimina los mensajes del buzon actual despues de copiarlos

imap_mail_move

(PHP 3, PHP 4 , PHP 5)

imap_mail_move -- Mueve los mensajes especificados a un buzón

Descripción

int **imap_mail_move** (int imap_stream, string msglist, string mbox)

Mueve los mensajes especificados por *msglist* al buzón especicifado. *msglist* es un rango no números de mensajes.

Devuelve **`TRUE`** si no hay error y **`FALSE`** en caso contrario.

imap_mail

(PHP 3>= 3.0.14, PHP 4 , PHP 5)

imap_mail -- Send an email message

Description

bool **imap_mail** (string to, string subject, string message [, string additional_headers [, string cc [, string bcc [, string rpath]]]])

This function allows sending of emails with correct handling of Cc and Bcc receivers. Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo..

The parameters *to*, *cc* and *bcc* are all strings and are all parsed as rfc822 address lists.

The receivers specified in *bcc* will get the mail, but are excluded from the headers.

Use the *rpath* parameter to specify return path. This is useful when using PHP as a mail client for multiple users.

imap_mailboxmsginfo

(PHP 3>= 3.0.2, PHP 4 , PHP 5)

imap_mailboxmsginfo -- Obtiene información acerca del buzón actual

Descripción

object **imap_mailboxmsginfo** (int imap_stream)

Devuelve información acerca del buzón actual. Devuelve **FALSE** en caso de fallo.

La función **imap_mailboxmsginfo()** comprueba el estado del buzón actual en el servidor y devuelve la información en un objeto con las siguientes propiedades.

Date : fecha del mensaje
Driver : driver
Mailbox : nombre del buzón
Nmsgs : número de mensajes
Recent : número de los mensajes recientes
Unread : número de los mensajes no leidos
Size : tamaÃ‚Â¤o del buzon

imap_mime_header_decode

(PHP 3>= 3.0.17, PHP 4 , PHP 5)

imap_mime_header_decode -- Decode MIME header elements

Description

array **imap_mime_header_decode** (string text)

imap_mime_header_decode() function decodes MIME message header extensions that are non ASCII text (see RFC2047) The decoded elements are returned in an array of objects, where each object has two properties, "charset" and "text". If the element hasn't been encoded, and in other words is in plain US-ASCII,the "charset" property of that element is set to "default".

Ejemplo 1. imap_mime_header_decode() example

```
<?php
$text = "=?ISO-8859-1?Q?Keld_J=F8rn_Simonsen?= <keld@example.com>";

$elements = imap_mime_header_decode($text);
for ($i=0; $i<count($elements); $i++) {
    echo "Charset: {$elements[$i]->charset}\n";
    echo "Text: {$elements[$i]->text}\n\n";
}
?>
```

In the above example we would have two elements, whereas the first element had previously been encoded with ISO-8859-1, and the second element would be plain US-ASCII.

imap_msgno

(PHP 3>= 3.0.3, PHP 4 , PHP 5)

imap_msgno -- Esta función devuelve el número de secuencia del mensaje para el UID dado.

Descripción

int **imap_msgno** (int imap_stream, int uid)

Esta función devuelve el número de secuencia del mensaje para el UID dado. Esta función es la inversa a **imap_uid()**.

imap_num_msg

(PHP 3, PHP 4 , PHP 5)

imap_num_msg -- Informa del número de mensajes en el buzón actual

Descripción

int **imap_num_msg** (int imap_stream)

Devuelve el número de mensajes en el buzón actual.

imap_num_recent

(PHP 3, PHP 4 , PHP 5)

imap_num_recent -- Informa el número de mensajes recientes en el buzón actual

Descripción

int **imap_num_recent** (int imap_stream)

Devuleve el número de mensajes recientes en el buzón actual.

imap_open

(PHP 3, PHP 4 , PHP 5)

imap_open -- Abre una sesión IMAP

Descripción

int **imap_open** (string mailbox, string username, string password, int flags)

Devuleve la sesion IMAP si no hay error y **FALSE** en caso contrario. Esta función también puede ser usada para abrir sesiones con servidores POP3 y NNTP. Para conectarse a un servidor IMAP escuchando por el puerto 143 en una máquina local, haga lo siguiente:

```
$mbox = imap_open("{localhost:143}INBOX","user_id","password");
```

Para conectarse a un servidor POP3 escuchando por el puerto 110, use:

```
$mbox = imap_open("{localhost/pop3:110}INBOX","user_id","password");
```
Para conectarse a un servidor NNTP escuchando por el puerto 119, use:

```
$nntp = imap_open("{localhost/nntp:119}comp.test","","");
```
Para conectarse a un servidor remoto sustituya "localhost", por el nombre o dirección IP del servidor al cual quiere conectarse.

Las opciones son una máscara de bit con una o más de los siguientes:

- OP_READONLY - Abre el buzón en modo de sólo lectura
- OP_ANONYMOUS - No usa o actualiza un .newsrc para las noticias
- OP_HALFOPEN - Para nombres IMAP y NNTP, abre una conexión pero no abre un buzón
- CL_EXPUNGE - Purga automaticamente el buzón antes de cerrar la sesión

imap_ping

(PHP 3, PHP 4 , PHP 5)

imap_ping -- Comprueba si la sesión IMAP está aún activa

Descripción

int **imap_ping** (int imap_stream)

Devuelve **TRUE** si la sesión está activa, **FALSE** en caso contrario.

La función **imap_ping()** pings the stream to see it is still active. Esto puede descubrir que hay correo nuevo; este es el método preferido para hacer una comprobación periodica del buzón, asi como para mantener activa sesiones en servidores que tienen inactivity timeout.

imap_qprint

(PHP 3, PHP 4 , PHP 5)

imap_qprint -- Convierte una cadena quoted-printable a una cadena de 8 bit

Descripción

string **imap_qprint** (string string)

Convierte una cadena quoted-printable a una cadena de 8 bit

Devuelve una cadena de 8 bit (binary)

imap_renamemailbox

(PHP 3, PHP 4 , PHP 5)

imap_renamemailbox -- Renombra un buzón

Descripción

int **imap_renamemailbox** (int imap_stream, string old_mbox, string new_mbox)

Esta función renombra un buzon (ver **imap_open()** para el formato del parámetro *mbox*).

Devuelve **TRUE** si no hay error y **FALSE** en caso contrario.

Ver También **imap_createmailbox()** and **imap_deletemailbox()**.

imap_reopen

(PHP 3, PHP 4 , PHP 5)

imap_reopen -- Reabre una sesión IMAP a un nuevo buzón

Descripción

int **imap_reopen** (string imap_stream, string mailbox [, string flags])

Devuelve **TRUE** si no hay error y **FALSE** en caso contrario.

Esta funció reabre la sesión especificada con un nuevo buzón.

Las opciones son máscaras de bit con una o más de las siguientes:

- OP_READONLY - Abre el buzón en modo de sólo lectura
- OP_ANONYMOUS - No usa o actualiza .newsrc para noticias
- OP_HALFOPEN - Para nombres IMAP y NNTP, abre una conexión pero no abre el buzon.
- CL_EXPUNGE - Expurga automaticamente el buzon antes de cerra la sesión

imap_rfc822_parse_adrlist

(PHP 3>= 3.0.2, PHP 4 , PHP 5)

imap_rfc822_parse_adrlist -- Examina la cadena dirección

Descripción

string **imap_rfc822_parse_adrlist** (string address, string default_host)

Esta función examina la cadena dirección y para cada dirección, devuelve un array de objetos. Los 4 objetos son:

mailbox - el nombre del buzón (username)
host - el nombre del ordenador
personal - el nombre personal
adl - ruta del dominio

imap_rfc822_parse_headers

(PHP 4 , PHP 5)

imap_rfc822_parse_headers -- Parse mail headers from a string

Description

object **imap_rfc822_parse_headers** (string headers [, string defaulthost])

This function returns an object of various header elements, similar to **imap_header()**, except without the flags and other elements that come from the IMAP server.

imap_rfc822_write_address

(PHP 3>= 3.0.2, PHP 4 , PHP 5)

imap_rfc822_write_address -- Devuleve una dirección de correo correctamente formateada dado el buzón, host, e información personal.

Descripción

string **imap_rfc822_write_address** (string mailbox, string host, string personal)

Devuelve una dirección de correo correctamente formateada, dado el buzón, host, e información personal.

imap_scanmailbox

(PHP 3, PHP 4 , PHP 5)

imap_scanmailbox -- Lee la lista de buzones y toma una cadena para buscar en el texto del buzón

Descripción

array **imap_scanmailbox** (int imap_stream, string string)

Devuelve un array que contiene los nombres de los buzones que tienen el parámetro *string* en el texto del buzón.

imap_search

(PHP 3>= 3.0.12, PHP 4 , PHP 5)

imap_search -- Esta función devuelve un array de mensajes que coinciden con el criterio de busqueda dado.

Descripción

array **imap_search** (int imap_stream, string criteria, int flags)

Esta función realiza una busqueda en el buzón actualmente abierto indicado por imap_stream. *criteria* es una cadena, delimitada por espacios, en la cual las siguientes palabras claves son permitidas. Cualquier argumento múltiple (ej. FROM "joey smith") debe estar entre comillas.

- ALL - devuelve todos los mensajes que coinciden con el resto del criterio
- ANSWERED - busca mensajes con el flag \\ANSWERED activado
- BCC "string" - busca mensajes con "cadena" en el campo Bcc:
- BEFORE "date" - busca mensajes con Date: antes de "date"
- BODY "string" - busca mensajes con "cadena" en el cuerpo del mensaje
- CC "string" - busca mensajes con "cadena" en el campo Cc:
- DELETED - busca mensajes eliminados
- FLAGGED - busca mensajes con el flag \\FLAGGED (sometimes referred to as Important or Urgent) activado
- FROM "string" - busca mensajes con "cadena" en el campo From:
- KEYWORD "string" - busca mensajes con "cadena" como una palabra clave
- NEW - busca mensajes nuevos
- OLD - busca mensajes viejos
- ON "date" - busca mensajes con "date" igual a Date:
- RECENT - busca mensajes con el flag \\RECENT activado
- SEEN - busca mensajes que han sido leidos (la opcion \\SEEN activada)
- SINCE "date" - busca mensajes conwith Date: after "date"
- SUBJECT "string" - busca mensajes con "string" en el campo Subject:
- TEXT "string" - busca mensajes con el texto "string"

- TO "string" - busca mensajes con "string" en el campo To:
- UNANSWERED - busca mensajes que no han sido respondidos
- UNDELETED - busca mensajes que no han sido eliminados
- UNFLAGGED - busca mensajes que no estan flagged
- UNKEYWORD "string" - busca mensajes que no coinciden con la palabra clave "string"
- UNSEEN - busca mensajes que no han sido leidos aún

Por ejemplo, para buscar todos los mensajes no contestados enviados por Mamá, usaría: "UNANSWERED FROM mamá". La busqueda parece ser no sensitiva. Esta lista de criterios está tomada del codigo fuente del UW c-client y puede que este incompleta o sea inexacta.

Valores validos para los flags son SE_UID, que provoca que el array devuelto contenga UIDs en vez de los numeros de secuencia de los mensajes

imap_set_quota

(PHP 4 >= 4.0.5, PHP 5)

imap_set_quota -- Sets a quota for a given mailbox

Description

bool **imap_set_quota** (resource imap_stream, string quota_root, int quota_limit)

Sets an upper limit quota on a per mailbox basis. This function requires the *imap_stream* to have been opened as the mail administrator account. It will not work if opened as any other user.

This function is currently only available to users of the c-client2000 or greater library.

imap_stream is the stream pointer returned from a **imap_open()** call. This stream must be opened as the mail administrator, other wise this function will fail. *quota_root* is the mailbox to have a quota set. This should follow the IMAP standard format for a mailbox, 'user.name'. *quota_limit* is the maximum size (in KB) for the *quota_root*.

Returns **TRUE** on success and **FALSE** on error.

Ejemplo 1. imap_set_quota() example

```
<?php
$mbox = imap_open("{your.imap.host:143}", "mailadmin", "password");

if (!imap_set_quota($mbox, "user.kalowsky", 3000)) {
    echo "Error in setting quota\n";
    return;
}

imap_close($mbox);
?>
```

See also **imap_open()** and **imap_set_quota()**.

imap_setacl

(PHP 4 >= 4.1.0, PHP 5)

imap_setacl -- Sets the ACL for a giving mailbox

Description

bool **imap_setacl** (resource stream_id, string mailbox, string id, string rights)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

This function is currently only available to users of the c-client2000 or greater library.

See also **imap_getacl()**.

imap_setflag_full

(PHP 3>= 3.0.3, PHP 4 , PHP 5)

imap_setflag_full -- Activa flags en los mensajes

Descripción

string **imap_setflag_full** (int stream, string sequence, string flag, string options)

Esta función aÃ‚Â¤ade el flag especificado al conjunto de flags activos para los mensajes en la secuencia especificada.

Los flags que puede seleccionar son "\\Seen", "\\Answered", "\\Flagged", "\\Deleted", "\\Draft", y "\\Recent" (definidos en el RFC2060)

Las opciones son una máscara de bit con uno o más de los siguientes:

ST_UID El argumento sequence contiene UIDs en vez de números secuenciales

imap_sort

(PHP 3>= 3.0.3, PHP 4 , PHP 5)

imap_sort -- Ordena un array de cabeceras de mensajes

Descripción

string **imap_sort** (int stream, int criteria, int reverse, int options)

Devuelve un array de números de mensajes ordenados por los parametros dados

Rev es 1 para una ordenación inversa.

Criteria puede ser uno (y sólo uno) de los siguientes:

SORTDATE	Fecha del mensaje
SORTARRIVAL	Fecha de llegada
SORTFROM	mailbox in first From address
SORTSUBJECT	Asunto del mensaje
SORTTO	mailbox in first To address
SORTCC	mailbox in first cc address
SORTSIZE	tamaÃ,Â¤o del mensaje en bytes

Las opciones son una máscara de bit con uno o más de los siguientes:

SE_UID	Devuelve UIDs en vez de números secuenciales
SE_NOPREFETCH	No preselecciona los mensajes buscados.

imap_status

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

imap_status -- Esta función devuelve el información de estado de otro buzón distinto al actual.

Descripción

object **imap_status** (int imap_stream, string mailbox, int options)

Esta función devuelve un objeto que contiene información de estado. Las opciones válidas son:

- SA_MESSAGES - activa status->messages con el número de mensajes en el buzon
- SA_RECENT - activa status->recent con el número de mensajes recientes en el buzón
- SA_UNSEEN - activa status->unseen con el número de mesajes no leidos (nuevos) en el buzón
- SA_UIDNEXT - activa status->uidnext con el próximo uid a usar en el bizón
- SA_UIDVALIDITY - activa status->uidvalidity con una constante que cambia cuando los uids del buzon ya no son válidos
- SA_ALL - activa todos los de arriba

status->flags contienen una máscara de bits la cual puede ser comprobada contra cualquiera de las propiedades de arriba.

imap_subscribe

(PHP 3, PHP 4 , PHP 5)

imap_subscribe -- Subscribe to a mailbox

Descripción

int **imap_subscribe** (int imap_stream, string mbox)

Da de alta un nuevo buzón.

Devuleve **TRUE** si no hay error y **FALSE** en caso contrario.

imap_thread

(PHP 4 >= 4.1.0, PHP 5)

imap_thread -- Returns a tree of threaded message

Description

array **imap_thread** (resource stream_id [, int options])

imap_thread() returns an associative array containing a tree of messages threaded by *REFERENCES*, or **FALSE** on error.

Every message in the current mailbox will be represented by three entries in the resulting array:

- *$thread["XX.num"]* - current message number
- *$thread["XX.next"]*
- *$thread["XX.branch"]*

Ejemplo 1. imap_thread() Example

```php
<?php

// Here we're outputting the threads of a newsgroup, in HTML

$nntp = imap_open('{news.example.com:119/nntp}some.newsgroup', '', '');
$threads = imap_thread($nntp);

foreach ($thread as $key => $val) {
  $tree = explode('.', $key);
  if ($tree[1] == 'num') {
    $header = imap_headerinfo($nntp, $val);
    echo "<ul>\n\t<li>" . $header->fromaddress . "\n";
  } elseif ($tree[1] == 'branch') {
    echo "\t</li>\n</ul>\n";
  }
}

imap_close($nntp);

?>
```

imap_timeout

(PHP 4 >= 4.3.3, PHP 5)

imap_timeout -- Set or fetch imap timeout

Description

mixed **imap_timeout** (int timeout_type [, int timeout])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

imap_uid

(PHP 3>= 3.0.3, PHP 4 , PHP 5)

imap_uid -- Esta función devuelve el UID del número de secuencia del mensaje dado

Descripción

int **imap_uid** (int imap_stream, int msgno)

Esta función devuelve el UID del número de secuencia del mensaje dado. Esta función es la inversa a **imap_msgno()**.

imap_undelete

(PHP 3, PHP 4 , PHP 5)

imap_undelete -- Desmarca los mensajes que están marcados como borrados

Descripción

int **imap_undelete** (int imap_stream, int msg_number)

Esta funcion elimina la marca de borrado de un mensaje especifico, puesta por la función **imap_delete()**.

Devuleve **TRUE** si no hay error y **FALSE** en caso contrario.

imap_unsubscribe

(PHP 3, PHP 4 , PHP 5)

imap_unsubscribe -- Unsubscribe from a mailbox

Descripción

int **imap_unsubscribe** (int imap_stream, string mbox)

Da de baja el buzón especificado.

Devuelve **TRUE** si no hay error y **FALSE** en caso contrario.

imap_utf7_decode

(PHP 3>= 3.0.15, PHP 4 , PHP 5)

imap_utf7_decode -- Decodes a modified UTF-7 encoded string

Description

string **imap_utf7_decode** (string text)

Decodes modified UTF-7 *text* into ISO-8859-1 string.

Returns a string that is encoded in ISO-8859-1 and consists of the same sequence of characters in *text*, or **FALSE** if *text* contains invalid modified UTF-7 sequence or *text* contains a character that is not part of ISO-8859-1 character set.

This function is needed to decode mailbox names that contain certain characters which are not in range of printable ASCII characters.

The modified UTF-7 encoding is defined in RFC 2060, section 5.1.3 (original UTF-7 was defined in RFC1642).

See also: **imap_utf7_encode()**.

imap_utf7_encode

(PHP 3>= 3.0.15, PHP 4 , PHP 5)

imap_utf7_encode -- Converts ISO-8859-1 string to modified UTF-7 text

Description

string **imap_utf7_encode** (string data)

Converts *data* to modified UTF-7 text. Note that *data* is expected to be encoded in ISO-8859-1.

This is needed to encode mailbox names that contain certain characters which are not in range of printable ASCII characters.

The modified UTF-7 encoding is defined in RFC 2060, section 5.1.3 (original UTF-7 was defined in RFC1642).

See also: **imap_utf7_decode()**.

imap_utf8

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

imap_utf8 -- Converts MIME-encoded text to UTF-8

Description

string **imap_utf8** (string mime_encoded_text)

Converts the given *mime_encoded_text* to UTF-8. MIME encoding method and the UTF-8 specification are described in RFC2047 and RFC2044 respectively.

LIV. Opciones e Información de PHP

Introducción

Estas funciones le dan la capacidad de obtener una gran cantidad de información sobre PHP mismo, p.ej. configuración de tiempo de ejecución, extensiones cargadas, versiones y mucho más. También encontrará funciones para establecer opciones para su intérprete PHP en ejecución. La que es probablemente la función mejor conocida de PHP - **phpinfo()** - puede ser encontrada aquí.

Requirimientos

No se necesitan bibliotecas externas para construir esta extensión

---

Instalación

No se necesita ninguna instalación para usar estas funciones, son parte del núcleo de PHP.

Configuración en tiempo de ejecución

El comportamiento de estas funciones está afectado por los valores definidos en `php.ini`.

Tabla 1. Opciones de Configuración del sistema de información PHP

Nombre	Predeterminado	Modificable
assert.active	"1"	PHP_INI_ALL
assert.bail	"0"	PHP_INI_ALL
assert.warning	"1"	PHP_INI_ALL
assert.callback	NULL	PHP_INI_ALL
assert.quiet_eval	"0"	PHP_INI_ALL
enable_dl	"1"	PHP_INI_SYSTEM
max_execution_time	"30"	PHP_INI_ALL
max_input_time	"60"	PHP_INI_ALL
magic_quotes_gpc	"1"	PHP_INI_PERDIR| PHP_INI_SYSTEM
magic_quotes_runtime	"0"	PHP_INI_ALL

Para más detalles sobre las constantes PHP_INI_* y su definición, vea **ini_set()**.

A continuación se presenta una corta explicación de las directivas de configuración.

assert.active **boolean**

Habilitar la evaluación **assert()**.

assert.bail **boolean**

Terminar la ejecución del script cuando las aserciones fallen.

assert.warning **boolean**

Producir una advertencia PHP para cada aserción fallida.

assert.callback **string**

Función de usuario a llamar cuando las aserciones fallen.

assert.quiet_eval **boolean**

Usar el valor actual de **error_reporting()** durante la evaluación de expresiones asertivas. Si se habilita, no se muestran errores durante la evaluación (implicit error_reporting(0)). Si se

deshabilita, los errores son mostrados de acuerdo a los valores de **error_reporting()**.

enable_dl **boolean**

Esta directiva es útil en realidad únicamente en la versión de módulo de Apache de PHP. Puede habilitar y deshabilitar la carga dinámica de extensiones PHP con **dl()** para cada servidor virtual o por directorio.

La razón principal para deshabilitar la carga dinámica es la seguridad. Con la carga dinámica, es posible ignorar todas las restricciones open_basedir. El comportamiento predeterminado es permitir la carga dinámica, excepto cuando se usa safe mode. En safe mode, siempre es imposible usar **dl()**.

max_execution_time **integer**

Este valor define el tiempo máximo en segundos que se le permite correr a un script, antes de que sea detenido por el intérprete. Esto ayuda a prevenir que scripts pobremente escritos congestionen el servidor. El valor predeterminado es *30*.

El tiempo máximo de ejecución no es afectado por llamadas del sistema, operaciones de secuencias etc. Por favor consulte la función **set_time_limit()** para más detalles.

No puede modificar este parámetro con **ini_set()** cuando está corriendo bajo safe mode. El único modo de evitar este contratiempo es deshabilitar el modo seguro, o cambiar el tiempo límite en `php.ini`.

Su servidor web puede tener otros tiempos de espera. Por ejemplo, Apache tiene la directiva *Timeout*, IIS tiene la función timeout CGI, y ambos valores predeterminados son de 300 segundos. Vea la documentación del servidor web para consultar su significado.

max_input_time **integer**

Este valor establece el tiempo máximo en segundos con el que cuenta un script para recibir datos de entrada, como POST, GET y cargas de archivos. El valor predeterminado es *60*.

magic_quotes_gpc **boolean**

Establece el estado de magic_quotes para las operaciones GPC (Get/Post/Cookie). Cuando magic_quotes se encuentra activo, todos los caracteres ' (comilla-simple), " (comilla doble), \ (barra invertida) y NULs son escapados con una barra invertida automáticamente.

> **Nota:** Si la directiva magic_quotes_sybase se encuentra activa también, sobrescribirá completamente el valor de magic_quotes_gpc. Tener ambas directivas activas quiere decir que solo las comillas simples son escapadas como *''*. Las comillas dobles, las barras invertidas y NULs permanecerán intactos y sin escapar.

Vea también **get_magic_quotes_gpc()**.

magic_quotes_runtime **boolean**

Si *magic_quotes_runtime* está habilitado, la mayoría de funciones que devuelven datos de alguna clase de fuente externa, incluyendo bases de datos y archivos de texto, tendrán las comillas escapadas con barras invertidas. Si magic_quotes_sybase se encuentra habilitado

también, una comilla sencilla es escapada con una comilla sencilla en lugar de una barra invertida.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Las constantes listadas aqui están siempre disponibles a través del "núcleo PHP".

Tabla 2. Constantes predefinidas de phpcredits()

Constante	Valor	Descripción
CREDITS_GROUP	1	Una lista de los desarrolladores centrales
CREDITS_GENERAL	2	Créditos generales: Diseño y concepto del lenguaje, autores de PHP y del módulo SAPI.
CREDITS_SAPI	4	Una lista de los módulos API de servidor para PHP, y sus autores.
CREDITS_MODULES	8	Una lista de módulos de extensión de PHP, y sus autores.
CREDITS_DOCS	16	Los créditos para el equipo de documentación.
CREDITS_FULLPAGE	32	Usualmente usado en combinación con las otras banderas. Indica que es necesario imprimir una página HTML independiente que incluya la información indicada por las otras banderas.
CREDITS_QA	64	Los créditos para el equipo de revisión de calidad.
CREDITS_ALL	-1	Todos los créditos, ñalente a usar: CREDITS_DOCS + CREDITS_GENERAL + CREDITS_GROUP + CREDITS_MODULES + CREDITS_QA CREDITS_FULLPAGE. Genera una página HTML independiente completa con las etiquetas apropiadas. Este es el valor predeterminado.

Tabla 3. Constantes phpinfo()

Constante	Valor	Descripción
INFO_GENERAL	1	La línea de configuración, ubicación de `php.ini`, fecha de compilación, Servidor Web, Sistema y más.
INFO_CREDITS	2	Créditos PHP. Vea también **phpcredits()**.
INFO_CONFIGURATION	4	Valores Locales y Maestros actuales de las directivas PHP. Vea también **ini_get()**.
INFO_MODULES	8	Módulos cargados y sus respectivos parámetros.
INFO_ENVIRONMENT	16	Información de Variables de Entorno, que también está disponible en *$_ENV*.

Constante	Valor	Descripción
INFO_VARIABLES	32	Muestra todas las variables predefinidas de EGPCS (Entorno, GET, POST, Cookie, Servidor).
INFO_LICENSE	64	Información de la Licencia PHP. Vea también el faq de licencia.
INFO_ALL	-1	Muestra todo lo anterior. Este es el valor predeterminado.

ASSERT_ACTIVE (**integer**)

ASSERT_CALLBACK (**integer**)

ASSERT_BAIL (**integer**)

ASSERT_WARNING (**integer**)

ASSERT_QUIET_EVAL (**integer**)

Tabla de contenidos

php_uname -- Devuelve información sobre el sistema operativo en el que está corriendo PHP
phpcredits -- Imprime los créditos de PHP
phpinfo -- Recupera gran cantidad de información de PHP.
phpversion -- Obtiene la versión actual de PHP.
putenv -- Establece el valor de una variable de entorno.
restore_include_path -- Restablece el valor de la opción de configuración include_path
set_include_path -- Establece la opción de configuración include_path
set_magic_quotes_runtime -- Establece el valor de la configuración activa actual de magic_quotes_runtime.
set_time_limit -- limita el tiempo máximo de ejecución
version_compare -- Compara dos cadenas de número de versión "PHP-estándar"
zend_logo_guid -- Obtiene el guid zend
zend_version -- Obtiene la versión del motor Zend actual

assert_options

(PHP 4 , PHP 5)

assert_options -- Establecer/obtener las varias banderas de aserción

Descripción

mixed **assert_options** (int que [, mixed valor])

Mediante el uso de **assert_options()**, usted puede establecer las varias opciones de control de **assert ()**, o simplemente consultar sus valores actuales.

Tabla 1. Opciones de Assert

opción	parámetro-ini	predeterminado	descripción
ASSERT_ACTIVE	assert.active	1	habilitar la evaluación **assert()**
ASSERT_WARNING	assert.warning	1	generar una advertencia PHP para cada aserción fallida
ASSERT_BAIL	assert.bail	0	terminar la ejecución cuando las aserciones fallen
ASSERT_QUIET_EVAL	assert.quiet_eval	0	deshabilitar error_reporting durante la evaluación de expresiones de aserción
ASSERT_CALLBACK	assert.callback	(**NULL**)	función de usuario a llamar cuando las aserciones fallen

assert_options() devolverá el valor original de cualquier opción, o **FALSE** si ocurren fallos.

assert

(PHP 4 , PHP 5)

assert -- Revisa si la aserción es evaluada a **FALSE**

Descripción

int **assert** (mixed asercion)

assert() revisará la *asercion* dada y tomará una acción apropiada si su resultado es **FALSE**.

Si la *asercion* es dada como una cadena, ésta será evaluada como código PHP por **assert()**. Las ventajas de una *asercion* tipo cadena son menor sobrecarga cuando el chequeo de aserciones se encuentre deshabilitado, y la producción de mensajes que contengan la expresión *asercion* cuando ésta falle. Esto quiere decir que si pasa una condición booleana como *asercion*, ésta condición no aparecerá como parámetro de la función de aserción, la cual pudo haber definido con la función **assert_options()**, la condición es convertida en una cadena antes de llamar tal función gestora, y el valor booleano **FALSE** es convertido como una cadena vacía.

Las aserciones deben ser usadas como una característica de depuración únicamente. Puede usarlas para realizar chequeos de integridad que prueben condiciones que deberían ser siempre **TRUE** y que indiquen algunos errores de programación si no se cumplen, o que chequeen por la presencia de ciertas características como las funciones de extensión, o ciertos límites y características de sistema.

Las aserciones no deberían ser usadas para operaciones normales de tiempo de ejecución como chequeos de parámetros de entrada. Como regla de oro, su código debería trabajar correctamente siempre si el chequeo de aserciones no está activado.

El comportamiento de **assert()** puede ser configurado por **assert_options()** o mediante los parámetros .ini descritos en su respectiva página del manual.

La función **assert_options()** o la directiva de configuración ASSERT_CALLBACK permiten el uso de llamadas de retorno para gestionar las aserciones fallidas.

Las llamadas de retorno de **assert()** son particularmente útiles para productos que prueben procesos automatizados de generación, ya que le permitin capturar fácilmente el código pasado a la aserción, junto con información sobre dónde se creó la aserción. Aunque esta información puede ser capturada mediante otros métodos, ¡el uso de aserciones lo hace mucho más rápido y sencillo!

La llamada de retorno debe aceptar tres argumentos. El primer argumento contendrá el archivo en el que ha fallado la aserción. El segundo argumento contendrá la línea en la que falló la aserción, y el tercer argumento contendrá la expresión que falló (si está disponible - valores literales como 1 o "dos" no serán pasados a través de este argumento).

Ejemplo 1. Gestionar una aserción fallida con una función personalizada

```
<?php
// Activar las aserciones y hacerlas calladas
assert_options(ASSERT_ACTIVE, 1);
assert_options(ASSERT_WARNING, 0);
assert_options(ASSERT_QUIET_EVAL, 1);

// Crear una funcion gestora
function mi_gestor_de_asercion($archivo, $linea, $codigo)
{
    echo "<hr>Aserci&oacute;n Fallida:
        Archivo '$archivo'<br />
        Linea   '$linea'<br />
        Codigo  '$codigo'<br /><hr />";
}

// Configurar la llamada de retorno
assert_options(ASSERT_CALLBACK, 'mi_gestor_de_asercion');

// Crear una aserci&oacute;n que deber&iacute;a fallar
assert('mysql_query ("")');
?>
```

dl

(PHP 3, PHP 4 , PHP 5)

dl -- Carga una extensión PHP en tiempo de ejecución

Descripción

int **dl** (string biblioteca)

Carga la extensión PHP dada por el parámetro *biblioteca*. El parámetro *biblioteca* es *únicamente* el nombre de archivo de la extensión a cargar, el cual también depende de su plataforma. Por ejemplo, la extensión sockets (si fue compilada como módulo, ¡que no es el comportamiento predeterminado!) podría llamarse `sockets.so` en plataformas Unix, mientras que se llama `php_sockets.dll` en la plataforma windows.

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo. Si la funcionalidad de carga de módulos no está disponible (ver Nota) o ha sido deshabilitada (ya sea mediante la desactivación de *enable_dl* o activando safe mode en `php.ini`) un **`E_ERROR`** es producido y se detiene la ejecución. Si **dl()** falla porque la biblioteca especificada no pudo ser cargada, se emite un mensaje **`E_WARNING`** en compañía del **`FALSE`**.

Use **extension_loaded()** para probar si una cierta extensión ya se encuentra disponible o no. Esto funciona tanto con extensiones integradas como con las cargadas dinámicamente (ya sea mediante `php.ini` o **dl()**).

La función **dl()** es obsoleta a partir de PHP 5. Use el método de las Directivas de Carga de Extensiones en su lugar.

Ejemplo 1. Ejemplos de dl()

```
<?php
// Ejemplo de carga de una extension en base al SO
if (!extension_loaded('sqlite')) {
    if (strtoupper(substr(PHP_OS, 0, 3) == 'WIN')) {
        dl('php_sqlite.dll');
    } else {
        dl('sqlite.so');
    }
}

// O, usando la constante PHP_SHLIB_SUFFIX que esta disponible a
// partir de PHP 4.3.0
if (!extension_loaded('sqlite')) {
    $prefijo = (PHP_SHLIB_SUFFIX == 'dll') ? 'php_' : '';
    dl($prefijo . 'sqlite.' . PHP_SHLIB_SUFFIX);
}
?>
```

El directorio desde donde es cargada la extensión depende de su plataforma:

Windows - Si no está definida explícitamente en `php.ini`, la extensión es cargada desde `c:\php4\extensions\` por defecto.

Unix - Si no está definida explícitamente en `php.ini`, el directorio de extensiones predeterminado depende de

- si PHP ha sido compilado con *--enable-debug* o no
- si PHP ha sido compilado con soporte ZTS (Zend Thread Safety) (experimental) o no
- el valor interno *ZEND_MODULE_API_NO* actual (número interno de API de módulo Zend, el cual es básicamente la fecha en la que ocurrié un cambio significativo en la API de módulo, p.ej. *20010901*)

Tomando en cuenta lo anterior, el directorio recibe el valor predeterminado de *<dir-instalacion>/lib/php/extensions/ <debug-o-no>-<zts-o-no>-ZEND_MODULE_API_NO*, p.ej. `/usr/local/php/lib/php/extensions/debug-non-zts-20010901` o `/usr/local/php/lib/php/extensions/no-debug-zts-20010901`.

> **Nota:** La función **dl()** *no* es soportada en servidores Web multi-hilos. Use la sentencia *extensions* en su `php.ini` cuando trabaje sobre ese tipo de entornos. Sin embargo, ¡las versiones *CGI* y *CLI* **no** son afectadas!
>
> **Nota: dl()** es sensible a mayúsculas y minúsculas en plataformas Unix.
>
> **Nota:** Esta función no está habilitada en safe-mode (modo-seguro)

Vea también Directivas de Carga de Extensión y **extension_loaded()**.

extension_loaded

(PHP 3>= 3.0.10, PHP 4 , PHP 5)

extension_loaded -- averigua si una extensión ha sido cargada

Descripción

bool **extension_loaded** (string name)

Devuelve **TRUE** si la extensión identificada por *name* (nombre) está cargada. Puede ver el nombre de varias extensiones utilizando **phpinfo()**.

Véase también **phpinfo()**.

> **Nota:** Esta función fue añadida en 3.0.10.

get_cfg_var

(PHP 3, PHP 4 , PHP 5)

get_cfg_var -- Obtiene el valor de una opción de configuración de PHP

Descripción

string **get_cfg_var** (string nombre_var)

Devuelve el valor actual de la opción de configuración de PHP indicada por *nombre_var*, o **FALSE** si ocurre un error.

No devolverá información de configuración definida cuando PHP fuera compilado, ni leer desde un archivo de configuración de Apache (usando las directivas php3_configuration_option).

Para chequear si el sistema está usando un archivo de configuración, intente recuperar el valor del parámetro de configuración cfg_file_path. Si éste se encuentra disponible, quiere decir que un archivo de configuración está siendo usado.

Vea también **ini_get()**.

get_current_user

(PHP 3, PHP 4 , PHP 5)

get_current_user -- Obtiene el nombre del propietario del script PHP actual.

Descripción

string **get_current_user** (void)

Devuelve el nombre del propietario del script PHP actual.

Véase también **getmyuid()**, **getmypid()**, **getmyinode()**, y **getlastmod()**.

get_defined_constants

(PHP 4 >= 4.1.0, PHP 5)

get_defined_constants -- Devuelve una matriz asociativa con los nombres de todas las constantes y sus valores

Descripción

array **get_defined_constants** ([mixed categorizar])

Esta función devuelve los nombres y valores de todas las constantes definidas actualmente. Esto incluye aquellas creadas por extensiones, asi como aquellas creadas con la función **define()**.

Por ejemplo, la siguiente línea:

```
<?php
print_r(get_defined_constants());
?>
```

imprimirá una lista como:

```
Array
(
    [E_ERROR] => 1
    [E_WARNING] => 2
    [E_PARSE] => 4
    [E_NOTICE] => 8
    [E_CORE_ERROR] => 16
    [E_CORE_WARNING] => 32
    [E_COMPILE_ERROR] => 64
    [E_COMPILE_WARNING] => 128
    [E_USER_ERROR] => 256
    [E_USER_WARNING] => 512
    [E_USER_NOTICE] => 1024
    [E_ALL] => 2047
    [TRUE] => 1
)
```

A partir de PHP 5, un parámetro adicional, *categorizar*, puede ser pasado, causando que esta función devuelva una matriz multi-dimensional con categorías en las claves de la primera dimensión y constantes y sus valores en la segunda dimensión.

```
<?php
define("MI_CONSTANTE", 1);
print_r(get_defined_constants(true));
?>
```

El resultado del ejemplo seria algo similar a:

```
Array
(
    [internal] => Array
        (
            [E_ERROR] => 1
            [E_WARNING] => 2
            [E_PARSE] => 4
            [E_NOTICE] => 8
            [E_CORE_ERROR] => 16
            [E_CORE_WARNING] => 32
            [E_COMPILE_ERROR] => 64
            [E_COMPILE_WARNING] => 128
            [E_USER_ERROR] => 256
            [E_USER_WARNING] => 512
            [E_USER_NOTICE] => 1024
            [E_ALL] => 2047
            [TRUE] => 1
        )

    [pcre] => Array
        (
            [PREG_PATTERN_ORDER] => 1
            [PREG_SET_ORDER] => 2
            [PREG_OFFSET_CAPTURE] => 256
            [PREG_SPLIT_NO_EMPTY] => 1
            [PREG_SPLIT_DELIM_CAPTURE] => 2
            [PREG_SPLIT_OFFSET_CAPTURE] => 4
            [PREG_GREP_INVERT] => 1
        )

    [user] => Array
        (
            [MI_CONSTANTE] => 1
        )

)
```

Nota: El valor del parámetro *categorizar* es irrelevante, sólo se considera su presencia.

Vea también **defined()**, **get_loaded_extensions()**, **get_defined_functions()**, y **get_defined_vars()**.

get_extension_funcs

(PHP 4 , PHP 5)

get_extension_funcs -- Devuelve una matriz con los nombres de funciones de un módulo

Descripción

array **get_extension_funcs** (string nombre_modulo)

Esta función devuelve los nombres de todas las funciones definidas en el módulo indicado por *nombre_modulo*.

Nota: El parámetro *nombre_modulo* debe estar en *minúsculas*.

Por ejemplo, las siguientes líneas

```
<?php
print_r(get_extension_funcs("xml"));
print_r(get_extension_funcs("gd"));
?>
```

imprimirán una lista de las funciones en los módulos *xml* y *gd* respectivamente.

Vea también: **get_loaded_extensions()**

get_include_path

(PHP 4 >= 4.3.0, PHP 5)

get_include_path -- Obtiene la opción de configuración include_path actual

Descripción

string **get_include_path** (void)

Obtiene el valor actual de la opción de configuración include_path.

Ejemplo 1. Ejemplo de get_include_path()

```
<?php
// Funciona a partir de PHP 4.3.0
echo get_include_path();

// Funciona en todas las versiones de PHP
echo ini_get('include_path');
?>
```

Vea también **ini_get()**, **restore_include_path()**, **set_include_path()**, y **include()**.

get_included_files

(PHP 4 , PHP 5)

get_included_files -- Devuelve una matriz con los nombres de los archivos incluidos o requeridos

Descripción

array **get_included_files** (void)

Devuelve una matriz de nombres de todos los archivos que han sido incluidos usando **include()**, **include_once()**, **require()** o **require_once()**.

El script llamado originalmente es considerado un "archivo incluido", así que será listado junto con los archivos referenciados por la familia de funciones **include()**.

Los archivos que son incluidos o requeridos múltiples veces solo aparecen una vez en la matriz devuelta.

> **Nota:** Los archivos incluidos usando la directiva de configuración *auto_prepend_file* no se incluyen en la matriz devuelta.

Ejemplo 1. Ejemplo de get_included_files() (`abc.php`)

```php
<?php

include 'test1.php';
include_once 'test2.php';
require 'test3.php';
require_once 'test4.php';

$archivos_incluidos = get_included_files();

foreach ($archivos_incluidos as $nombre_archivo) {
    echo "$nombre_archivo\n";
}

?>
```

generará la siguiente salida:

```
abc.php
test1.php
test2.php
test3.php
test4.php
```

> **Nota:** En PHP 4.0.1pl2 y versiones anteriores, **get_included_files()** asumía que los archivos requeridos finalizaban en la extensión *.php*; otras extensiones no serán devueltas. La matriz devuelta por **get_included_files()** era una matriz asociativa y solo listaba los archivos incluidos por **include()** y **include_once()**.

Vea también **include()**, **include_once()**, **require()**, **require_once()**, y **get_required_files()**.

get_loaded_extensions

(PHP 4 , PHP 5)

get_loaded_extensions -- Devuelve una matriz con los nombres de todos los módulos compilados y cargados

Descripción

array **get_loaded_extensions** (void)

Esta función devuelve los nombres de todos los módulos compilados y cargados en el intérprete PHP.

Por ejemplo, la siguiente línea

```php
<?php
print_r(get_loaded_extensions());
?>
```

imprimirá una lista como:

```
Array
(
    [0] => xml
    [1] => wddx
    [2] => standard
    [3] => session
    [4] => posix
    [5] => pgsql
    [6] => pcre
    [7] => gd
    [8] => ftp
    [9] => db
    [10] => calendar
    [11] => bcmath
)
```

Vea también **get_extension_funcs()**, **extension_loaded()**, **dl()**, y **phpinfo()**.

get_magic_quotes_gpc

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

get_magic_quotes_gpc -- Obtiene el valor de la configuración activa actual de las comillas mágicas gpc.

Descripción

long **get_magic_quotes_gpc** (void)

Devuelve el valor de la configuración activa actual de magic_quotes_gpc. (0 desactivado, 1 activado)

Véase también **get_magic_quotes_runtime()**, **set_magic_quotes_runtime()**.

get_magic_quotes_runtime

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

get_magic_quotes_runtime -- Obtiene el valor de la configuración activa actual de magic_quotes_runtime.

Descripción

long **get_magic_quotes_runtime** (void)

Devuelve el valor de la configuración activa actual de magic_quotes_runtime. (0 desactivado, 1 activado)

Véase también **get_magic_quotes_gpc()**, **set_magic_quotes_runtime()**.

get_required_files

get_required_files -- Alias de **get_included_files()**

Descripción

Esta función es un alias de **get_included_files()**.

getenv

(PHP 3, PHP 4 , PHP 5)

getenv -- Obtiene el valor de una variable de entorno

Descripción

string **getenv** (string varname)

Devuelve el valor de la variable de entorno *varname*, o **FALSE** en caso de error.

```
$ip = getenv("REMOTE_ADDR"); // get the ip number of the user
```

Puede ver una lista de todas las variables de entorno utilizando **phpinfo()**. Puede encontrar el significado de la mayoría echando un vistazo en CGI specification (especificación CGI), especialmente en page on environmental variables (página de variables de entorno).

getlastmod

(PHP 3, PHP 4 , PHP 5)

getlastmod -- Recupera la fecha/hora de la última modificación de la página.

Descripción

int **getlastmod** (void)

Devuelve la fecha/hora de la última modificación de la página actual. El valor devuelto está en formato de fecha/hora Unix, adecuado para que sirva a **date()**. Devuelve **FALSE** en caso de error.

Ejemplo 1. ejemplo getlastmod()

```
// outputs e.g. 'Last modified: March 04 1998 20:43:59.'
echo "Last modified: ".date( "F d Y H:i:s.", getlastmod() );
```

Véase también **date()**, **getmyuid()**, **get_current_user()**, **getmyinode()**, y **getmypid()**.

getmygid

(PHP 4 >= 4.1.0, PHP 5)

getmygid -- Obtener el GID del dueño del script PHP

Descripción

int **getmygid** (void)

Devuelve el ID de grupo del script actual, o **FALSE** en caso de fallo.

Vea también **getmyuid()**, **getmypid()**, **get_current_user()**, **getmyinode()**, y **getlastmod()**.

getmyinode

(PHP 3, PHP 4 , PHP 5)

getmyinode -- Recupera el inodo del script actual.

Descripción

int **getmyinode** (void)

Devuelve el inodo del script actual, o **FALSE** en caso de error.

Véase también **getmyuid()**, **get_current_user()**, **getmypid()**, y **getlastmod()**.

getmypid

(PHP 3, PHP 4 , PHP 5)

getmypid -- Obtiene el ID de proceso de PHP.

Descripción

int **getmypid** (void)

Devuelve el ID del proceso PHP actual, o **FALSE** en caso de error.

Advierta que cuando se ejecuta como un módulo de servidor, diferentes llamadas del script no garantizan que tengan distintos pids.

Véase también **getmyuid()**, **get_current_user()**, **getmyinode()**, y **getlastmod()**.

getmyuid

(PHP 3, PHP 4 , PHP 5)

getmyuid -- Obtiene el UID del propietario del script PHP.

Descripción

int **getmyuid** (void)

Devuelve el ID de usuario del script actual, o **FALSE** en caso de error.

Véase también **getmypid()**, **get_current_user()**, **getmyinode()**, y **getlastmod()**.

getopt

(PHP 4 >= 4.3.0, PHP 5)

getopt -- Obtiene opciones de la lista de argumentos desde la línea de comandos

Descripción

array **getopt** (string opciones [, array opc_largas])

Devuelve una matriz asociativa con parejas opción / argumento en base al formato de opciones especificado en *opciones*, o **FALSE** en caso de fallo.

En plataformas que tienen la función de C getopt_long, pueden especificarse opciones largas con el parámetro *opc_largas* (a partir de PHP 4.3.0).

```
<?php
// procesar la linea de comandos ($GLOBALS['argv'])
$opciones = getopt("f:hp:");
?>
```

El parámetro *opciones* puede contener los siguientes elementos: caracteres individuales, y caracteres seguidos de dos puntos para indicar que sigue un argumento que declara una opción. Por ejemplo, la cadena *x* reconoce una opción *-x*, y una cadena *x:* reconoce una opción y un argumento *-x argumento*. No importa si un argumento incluye espacio en blanco al comienzo.

Esta función devolverá una matriz de parejas opción / argumento. Si una opción no tiene argumento, el valor será definido a **FALSE**.

> **Nota:** Esta función no está implementada en plataformas Windows.

getrusage

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

getrusage -- Obtiene el consumo actual de recursos.

Descripción

array **getrusage** ([int who])

Es un interface a getrusage(2). Devuelve un array asociativo que contiene los datos devueltos de la llamada del sistema. Si who (quien) es 1, getrusage debería llamarse con RUSAGE_CHILDREN. Todas las entradas son accesibles utilizando sus nombres de campo documentados.

Ejemplo 1. Ejemplo Getrusage

```
$dat = getrusage();
echo $dat["ru_nswap"];          # number of swaps
echo $dat["ru_majflt"];         # number of page faults
echo $dat["ru_utime.tv_sec"];   # user time used (seconds)
echo $dat["ru_utime.tv_usec"]; # user time used (microseconds)
```

Vea la página man de system para más detalles.

ini_alter

ini_alter -- Alias de **ini_set()**

Descripción

Esta función es un alias de **ini_set()**.

ini_get_all

(PHP 4 >= 4.2.0, PHP 5)

ini_get_all -- Obtiene todas las opciones de configuración

Descripción

array **ini_get_all** ([string extension])

Devuelve todas las opciones de configuración registradas como una matriz asociativa. Si el parámetro opcional *extension* está definido, devuelve solo las opciones específicas para esa extensión.

La matriz devuelta usa el nombre de directiva como clave de la matriz, mientras los elementos de esa matriz son *global_value* (definido en `php.ini`), *local_value* (quizás definido con **ini_set()** o en `.htaccess`), y *access* (el nivel de acceso). Vea la sección del manual sobre cambios de configuración para más información sobre qué quieren decir los niveles de acceso.

> **Nota:** Es posible que una directiva tenga múltiples niveles de acceso, razón por la cual *access* muestra los valores de máscara de bits apropiados.

Ejemplo 1. Un ejemplo de ini_get_all()

```
<?php
$inis = ini_get_all();

print_r($inis);


?>
```

La salida parcial podría verse como:

```
Array
(
    [allow_call_time_pass_reference] => Array
    (
        [global_value] => 1
        [local_value] => 1
        [access] => 6
    )
    [allow_url_fopen] => Array
    (
        [global_value] => 1
        [local_value] => 1
        [access] => 7
    )

    ...

)
```

Vea también: **ini_get()**, **ini_restore()**, **ini_set()**, **get_loaded_extensions()**, y **phpinfo()**.

ini_get

(PHP 4 , PHP 5)

ini_get -- Obtiene el valor de una opción de configuración

Descripción

string **ini_get** (string nombre_var)

Devuelve el valor de la opción de configuración en caso de éxito. Si ocurre un fallo, como que se realice una consulta por un valor que no existe, devolverá una cadena vacía.

> **Cuando se consultan valores booleanos:** Un valor ini booleano de *off* será devuelto como una cadena vacía o "0", mientras que un valor ini booleano de *on* será devuelto como "1".
>
> **Cuando se consultan valores de tamaño de memoria:** Muchos valores ini de tamaño de memoria, como upload_max_filesize son almacenados en el archivo `php.ini` en notación abreviada. **ini_get()** devolverá la cadena exacta almacenada en el archivo `php.ini` y *NO* su ñalente **integer**. Usar funciones aritméticas normales sobre éstos valores no tendrían los resultados que de otra forma podrían esperarse. El siguiente ejemplo muestra una manera de convertir la notación corta de bytes, de forma similar al modo que lo hace el código fuente de PHP.

Ejemplo 1. Algunos ejemplos de ini_get()

```
<?php
/*
Nuestro php.ini contiene los siguientes parametros:

display_errors = On
register_globals = Off
post_max_size = 8M
*/

echo 'display_errors = ' . ini_get('display_errors') . "\n";
echo 'register_globals = ' . ini_get('register_globals') . "\n";
echo 'post_max_size = ' . ini_get('post_max_size') . "\n";
echo 'post_max_size+1 = ' . (ini_get('post_max_size')+1) . "\n";
echo 'post_max_size en bytes = ' . return_bytes(ini_get('post_max_size'));

function return_bytes($val) {
    $val = trim($val);
    $ultimo = $val{strlen($val)-1};
    switch($ultimo) {
        case 'k':
        case 'K':
            return (int) $val * 1024;
            break;
        case 'm':
        case 'M':
            return (int) $val * 1048576;
            break;
        default:
            return $val;
    }
}

?>
```

Este script producirá:

```
display_errors = 1
register_globals = 0
post_max_size = 8M
post_max_size+1 = 9
post_max_size en bytes = 8388608
```

Vea también **get_cfg_var()**, **ini_get_all()**, **ini_restore()**, y **ini_set()**.

ini_restore

(PHP 4 , PHP 5)

ini_restore -- Restablece el valor de una opción de configuración

Descripción

void **ini_restore** (string nombre_var)

Restablece una opción de configuración dada a su valor original.

Vea también **ini_get()**, **ini_get_all()**, y **ini_set()**.

ini_set

(PHP 4 , PHP 5)

ini_set -- Establece el valor de una opción de configuración

Descripción

string **ini_set** (string nombre_var, string nuevo_valor)

Establece el valor de la opción de configuración dada. Devuelve el valor antiguo en caso de éxito, **FALSE** en caso de fallo. La opciÃƒâ€œn de configuración mantendrá este nuevo valor durante la ejecución del script, y será restablecido al final del script.

No todas las opciones disponibles pueden ser modificadas usando **ini_set()**. Hay una lista de todas las opciones disponibles en el apéndice.

Vea también: **get_cfg_var()**, **ini_get()**, **ini_get_all()**, y **ini_restore()**

main

main -- Página predeterminada para **main()**

Descripción

No existe una función llamada **main()**, excepto en el código fuente de PHP. En PHP 4.3.0, un nuevo tipo de gestión de errores en las fuentes de PHP (php_error_docref) fue añadido. Una característica es la de proveer enlaces a la página del manual en los mensajes de error de PHP cuando las directivas PHP html_errors (habilitada por defecto) y docref_root (habilitada por defecto hasta PHP 4.3.2) están definidas.

Algunas veces los mensajes de error se refieren a una página del manual para la función **main()**, razón por la que existe esta página. Por favor agregue un comentario de usuario a continuación que mencione qué función de PHP causó el error que enlazaba con **main()** y ésta será arreglada y documentada apropiadamente.

Tabla 1. Errores conocidos que apuntan a main()

Nombre de función	Ya no apunta hacia aquí a partir de
include()	4.3.2
include_once()	4.3.2
require()	4.3.2
require_once()	4.3.2

Vea también html_errors y display_errors.

memory_get_usage

(PHP 4 >= 4.3.2, PHP 5)

memory_get_usage -- Devuelve la cantidad de memoria ubicada para PHP

Descripción

int **memory_get_usage** (void)

Devuelve la cantidad de memoria, en bytes, que está siendo ubicada actualmente para su script PHP.

memory_get_usage() se encontrará definida únicamente si su instalación de PHP es compilada con la opción de configuración *--enable-memory-limit*.

Ejemplo 1. Un ejemplo de memory_get_usage()

```
<?php
// Este es solo un ejemplo, los numeros presentados seran diferentes
// dependiendo de su sistema

echo memory_get_usage() . "\n"; // 36640

$a = str_repeat("Hola.", 4242);

echo memory_get_usage() . "\n"; // 57960

unset($a);

echo memory_get_usage() . "\n"; // 36744

?>
```

Vea también memory_limit.

php_ini_scanned_files

(PHP 4 >= 4.3.0, PHP 5)

php_ini_scanned_files -- Devolver una lista de archivos .ini procesados del directorio ini adicional

Descripción

string **php_ini_scanned_files** (void)

php_ini_scanned_files() devuelve una lista separada por comas de archivos de configuración procesados después de `php.ini`. Estos archivos se encuentran en un directorio definido por la opción *--with-config-file-scan-dir*, la cual es definida durante la compilación.

Devuelve una cadena, separada por comas, de archivos .ini en caso de éxito. Si la directiva *--with-config-files-scan-dir* no fue definida, se devuelve **`FALSE`**. Si la opción estaba definida y el directorio estaba vacío, se devuelve una cadena vacía. Si un archivo es irreconocible, éste aun será incluido en la cadena devuelta, pero también se producirá un error de PHP. Este error de PHP será visto tanto en tiempo de compilación como cuando se usa **php_ini_scanned_files()**.

Los archivos de configuración devueltos también incluyen la ruta, tal y como fuera sido declarada en la opción *--with-config-file-scan-dir*. Asimismo, cada coma es seguida por una nueva línea.

Ejemplo 1. Un ejemplo simple para listar los archivos ini devueltos

```
<?php
if ($lista_archivos = php_ini_scanned_files()) {
    if (strlen($lista_archivos) > 0) {
        $archivos = explode(',', $lista_archivos);

        foreach ($archivos as $archivo) {
            echo "<li>" . trim($archivo) . "</li>\n";
        }
    }
}
?>
```

Vea también **ini_set()** y **phpinfo()**.

php_logo_guid

(PHP 4 , PHP 5)

php_logo_guid -- Obtiene el guid logo

Descripción

string **php_logo_guid** (void)

> **Nota:** Esta funcionalidad fue añadida en PHP4 Beta 4.

php_sapi_name

(PHP 4 >= 4.0.1, PHP 5)

php_sapi_name -- Devuelve el tipo de interfaz entre el servidor web y PHP

Descripción

string **php_sapi_name** (void)

php_sapi_name() devuelve una cadena en minúsculas que describe el tipo de interfaz entre el servidor web y PHP (API de Servidor, SAPI). En PHP CGI, esta cadena es "cgi", en mod_php para Apache, esta cadena es "apache" y así sucesivamente.

Ejemplo 1. Ejemplo de php_sapi_name()

```
<?php
$tipo_sapi = php_sapi_name();
if (substr($tipo_sapi, 0, 3) == 'cgi') {
    echo "Est&aacute; usando PHP CGI\n";
} else {
    echo "No est&aacute; usando PHP CGI\n";
}
?>
```

php_uname

(PHP 4 >= 4.0.2, PHP 5)

php_uname -- Devuelve información sobre el sistema operativo en el que está corriendo PHP

Descripción

string **php_uname** ([string modo])

php_uname() devuelve una descripción del sistema operativo en el que PHP está corriendo. Para el nombre de tan solo el sistema operativo, considere usar la constante **PHP_OS**, pero recuerde que ésta constante contendrá el sistema operativo en el que PHP fue *compilado*.

En Unix, la salida recae a desplegar la información del sistema operativo en el que PHP fue compilado si no puede determinar el SO corriendo actualmente.

modo define la información a ser devuelta:

- *'a'*: Este es el valor predeterminado. Contiene todos los modo en la secuencia *"s n r v m"*.
- *'s'*: El nombre del sistema operativo. P.ej. *FreeBSD*.
- *'n'*: Nombre del host. P.ej. *localhost.example.com*.
- *'r'*: Nombre del lanzamiento. P.ej. *5.1.2-RELEASE*.
- *'v'*: Información de versión. Varía bastante entre sistemas operativos.
- *'m'*: Tipo de máquina. P.ej. *i386*.

Ejemplo 1. Algunos ejemplos de php_uname()

```
<?php
echo php_uname();
echo PHP_OS;

/* Algunas posibles salidas:
Linux localhost 2.4.21-0.13mdk #1 Fri Mar 14 15:08:06 EST 2003 i686
Linux

FreeBSD localhost 3.2-RELEASE #15: Mon Dec 17 08:46:02 GMT 2001
FreeBSD

Windows NT XN1 5.1 build 2600
WINNT
*/

if (strtoupper(substr(PHP_OS, 0, 3)) === 'WIN') {
    echo '&iexcl;Este es un servidor usando Windows!';
} else {
    echo '&iexcl;Este es un servidor que no usa Windows!';
}

?>
```

También existen algunas constantes de PHP predefinidas relacionadas que podrían resultar útiles, por ejemplo:

Ejemplo 2. Algunos ejemplos de constantes relacionadas con el SO

```
<?php
// *nix
echo DIRECTORY_SEPARATOR; // /
echo PHP_SHLIB_SUFFIX;    // so
echo PATH_SEPARATOR;      // :

// Win*
echo DIRECTORY_SEPARATOR; // \
echo PHP_SHLIB_SUFFIX;    // dll
echo PATH_SEPARATOR;      // ;
?>
```

Vea también **phpversion()**, **php_sapi_name()**, y **phpinfo()**.

phpcredits

(PHP 4 , PHP 5)

phpcredits -- Imprime los créditos de PHP

Descripción

void **phpcredits** ([int bandera])

Esta función imprime los créditos que listan los desarrolladores de PHP, los módulos, etc. Genera los códigos HTML apropiados para insertar la información en una página. *bandera* es opcional, y su valor predeterminado es **CREDITS_ALL**. Para generar una página de créditos personalizada, puede que quiera usar el parámetro *bandera*. Por ejemplo, para imprimir los créditos generales, usará algo así en alguna parte de su código:

```
<?php
phpcredits(CREDITS_GENERAL);
?>
```

Y si desea imprimir la lista de desarrolladores centrales y del grupo de documentación, en una página independiente, debe usar:

```
<?php
phpcredits(CREDITS_GROUP + CREDITS_DOCS + CREDITS_FULLPAGE);
?>
```

Y si desea embeber todos los créditos en su página, entonces algo como lo siguiente logrará el cometido:

```
<html>
 <head>
  <title>Mi p&aacute;gina de cr&eacute;ditos</title>
 </head>
 <body>
<?php
// algo de codigo suyo
phpcredits(CREDITS_ALL - CREDITS_FULLPAGE);
// mas codigo
?>
 </body>
</html>
```

Tabla 1. Banderas de phpcredits() predefinidas

nombre	descripción
CREDITS_ALL	Todos los créditos, ñalente a usar: CREDITS_DOCS + CREDITS_GENERAL + CREDITS_GROUP + CREDITS_MODULES + CREDITS_FULLPAGE. Genera una página HTML completa e independiente con las etiquetas apropiadas.
CREDITS_DOCS	Los créditos para el equipo de documentación
CREDITS_FULLPAGE	Usualmente usada en conjunto con las otras banderas. Indica que es necesario imprimir una página HTML completa e independiente, incluyendo la información indicada por las otras banderas.
CREDITS_GENERAL	Créditos generales: Diseño y concepto del lenguaje, autores de PHP 4.0 y módulo SAPI.
CREDITS_GROUP	Una lista de los desarrolladores centrales
CREDITS_MODULES	Una lista de los módulos de extensión para PHP, y sus autores
CREDITS_SAPI	Una lista de módulos API de servidor para PHP, y sus autores

Vea también **phpinfo()**, **phpversion()** y **php_logo_guid()**.

phpinfo

(PHP 3, PHP 4 , PHP 5)

phpinfo -- Recupera gran cantidad de información de PHP.

Descripción

int **phpinfo** (void)

Obtiene gran cantidad de información sobre el estado actual de PHP. Esto incluye información sobre las opciones de compilación y extensiones de PHP, la versión PHP, información y entorno del servidor (si está compilado como un módulo), el entorno PHP, información sobre la versión del SO, rutas, opciones de configuración maestras y locales, cabeceras HTTP, y la Licencia Pública GNU.

Véase también **phpversion()**.

phpversion

(PHP 3, PHP 4 , PHP 5)

phpversion -- Obtiene la versión actual de PHP.

Descripción

string **phpversion** (void)

Devuelve una cadena de carácteres que contiene la versión del parser PHP que está ejecutándose

actualmente.

Ejemplo 1. ejemplo phpversion()

```
// prints e.g. 'Current PHP version: 3.0rel-dev'
echo "Current PHP version: ".phpversion();
```

Véase también **phpinfo()**.

putenv

(PHP 3, PHP 4 , PHP 5)

putenv -- Establece el valor de una variable de entorno.

Descripción

void **putenv** (string setting)

Añade *setting (valor)* al entorno.

Ejemplo 1. Establecer una Variable de Entorno

```
putenv("UNIQID=$uniqid");
```

restore_include_path

(PHP 4 >= 4.3.0, PHP 5)

restore_include_path -- Restablece el valor de la opción de configuración include_path

Descripción

void **restore_include_path** (void)

Restablece la opción de configuración include_path de vuelta a su valor maestro original, tal y como se encuentre definido en `php.ini`

Ejemplo 1. Ejemplo de restore_include_path()

```
<?php

echo get_include_path();  // .:/usr/local/lib/php

set_include_path('/inc');

echo get_include_path();  // /inc

// Funciona a partir de PHP 4.3.0
restore_include_path();

// Funciona en todas las versiones de PHP
ini_restore('include_path');

echo get_include_path();  // .:/usr/local/lib/php

?>
```

Vea también **ini_restore()**, **set_include_path()**, **get_include_path()**, e **include()**.

set_include_path

(PHP 4 >= 4.3.0, PHP 5)

set_include_path -- Establece la opción de configuración include_path

Descripción

string **set_include_path** (string nueva_ruta_inclusion)

Establece la opción de configuración include_path durante la duración del script. Devuelve el valor antiguo de include_path en caso de éxito, o **FALSE** en caso de fallo.

Ejemplo 1. Ejemplo de set_include_path()

```
<?php
// Funciona a partir de PHP 4.3.0
set_include_path('/inc');

// Funciona en todas las versiones de PHP
ini_set('include_path', '/inc');
?>
```

Vea también **ini_set()**, **get_include_path()**, **restore_include_path()**, e **include()**.

set_magic_quotes_runtime

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

set_magic_quotes_runtime -- Establece el valor de la configuración activa actual de magic_quotes_runtime.

Descripción

long **set_magic_quotes_runtime** (int new_setting)

Establece el valor de la configuración activa actual de magic_quotes_runtime. (0 desactivado, 1 activado)

Véase también **get_magic_quotes_gpc()**, **get_magic_quotes_runtime()**.

set_time_limit

(PHP 3, PHP 4 , PHP 5)

set_time_limit -- limita el tiempo máximo de ejecución

Descripción

void **set_time_limit** (int seconds)

Establece el número de segundos que se le permite a un script ejecutarse. Si éste es alcanzado, el

script devuelve un error de tipo fatal. El límite por defecto es 30 segundos o, si existe, el valor *max_execution_time* definido en el `php.ini`. Si seconds (segundos) se establece a cero, no se impone ningún límite.

Cuando se llama, **set_time_limit()** reinicia el contador del timeout a cero. En otras palabras, si el timeout es el de por defecto de 30 segundos, y después de 25 segundos de ejecución del script se realiza una llamada set_time_limit(20), el script se ejecutará durante un total de 45 segundos antes de alcanzar su límite.

Aviso
Advierta que **set_time_limit()** no tiene efecto cuando PHP se ejecuta en safe mode (modo seguro). No hay otra opción que que desactivar el modo seguro o cambiar el límite de tiempo en el `php.ini`.

Nota: La función **set_time_limit()** y la directiva de configuración max_execution_time sólo afecta al tiempo de ejecución del script en sí. Cualquier tiempo consumido en una actividad que ocurre fuera de la ejecución del script así como llamadas al sistema utilizando la función **system()**, la función **sleep()**, consultas de base de datos, etc. no se incluye para determinar el tiempo máximo que lleva el script ejecutándose.

version_compare

(PHP 4 >= 4.1.0, PHP 5)

version_compare -- Compara dos cadenas de número de versión "PHP-estándar"

Descripción

int **version_compare** (string version1, string version2 [, string operador])

version_compare() compara dos cadenas de número de versión "PHP-estandarizadas". Esto es útil si quisiera escribir programas que trabajen solo en algunas versiones de PHP.

version_compare() devuelve -1 si la primera versión es inferior a la segunda, 0 si son iguales, y +1 si la segunda es menor.

La función comienza por reemplazar _, - y + con un punto . en las cadenas de versión, y también inserta puntos . antes y después de cualquier secuencia no numérica, de modo que por ejemplo '4.3.2RC1' se convierte en '4.3.2.RC.1'. Luego separa los resultados como si se usara explode('.', $ver). Luego compara las partes de izquierda a derecha. Si una parte contiene cadenas de versión especiales, éstas son gestionadas en el siguiente orden: *dev* < *alpha* = *a* < *beta* = *b* < *RC* < *pl*. De este modo, no solo las versiones con diferentes niveles, como '4.1' y '4.1.2' pueden ser comparadas, sino que también cualquier versión específica de PHP que contenga un indicador de estado de desarrollo.

Si especifica el tercer argumento opcional *operador*, puede realizar pruebas por una relación en particular. Los operadores posibles son: <, *lt*, <=, *le*, >, *gt*, >=, *ge*, ==, =, *eq*, *!=*, <>, *ne* respectivamente. Mediante el uso de este argumento, la función devolverá 1 si la relación es la especificada por el operador, o 0 de lo contrario.

Ejemplo 1. Ejemplo de version_compare()

```
<?php
// imprime -1
echo version_compare("4.0.4", "4.0.6");

// todos estos casos imprimen 1
echo version_compare("4.0.4", "4.0.6", "<");
echo version_compare("4.0.6", "4.0.6", "eq");
?>
```

zend_logo_guid

(PHP 4 , PHP 5)

zend_logo_guid -- Obtiene el guid zend

Descripción

string **zend_logo_guid** (void)

> **Nota:** Esta funcionalidad fue añadida en PHP4 Beta 4.

zend_version

(PHP 4 , PHP 5)

zend_version -- Obtiene la versión del motor Zend actual

Descripción

string **zend_version** (void)

Devuelve una cadena que contiene la versión del Motor Zend ejecutándose actualmente.

Ejemplo 1. Ejemplo de zend_version()

```
<?php
// imprime p.ej. 'Version del motor Zend: 1.0.4'
echo "Version del motor Zend: " . zend_version();
?>
```

Vea también **phpinfo()**, **phpcredits()**, **php_logo_guid()**, y **phpversion()**.

LV. Ingres II functions

Aviso
Esta extensión es *EXPERIMENTAL*. Esto significa que el comportamiento de esta extensión, los nombre de sus funciones y en definitiva TODO lo documentado sobre esta extensión, puede cambiar en una futura versión de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

These functions allow you to access Ingres II database servers.

In order to have these functions available, you must compile php with Ingres support by using the *--with-ingres* option. You need the Open API library and header files included with Ingres II. If the

II_SYSTEM environment variable isn't correctly set you may have to use *--with-ingres=DIR* to specify your Ingres installation directory.

When using this extension with Apache, if Apache does not start and complains with "PHP Fatal error: Unable to start ingres_ii module in Unknown on line 0" then make sure the environement variable II_SYSTEM is correctly set. Adding "export II_SYSTEM="/home/ingres/II" in the script that starts Apache, just before launching httpd, should be fine.

> **Nota:** If you already used PHP extensions to access other database servers, note that Ingres doesn't allow concurrent queries and/or transaction over one connection, thus you won't find any result or transaction handle in this extension. The result of a query must be treated before sending another query, and a transaction must be commited or rolled back before opening another transaction (which is automaticaly done when sending the first query).

Tabla de contenidos

ingres_autocommit

(PHP 4 >= 4.0.2, PHP 5)

ingres_autocommit -- Switch autocommit on or off.

Description

bool **ingres_autocommit** ([resource link])

ingres_autocommit() is called before opening a transaction (before the first call to **ingres_query()** or just after a call to **ingres_rollback()** or **ingres_autocommit()**) to switch the "autocommit" mode of the server on or off (when the script begins the autocommit mode is off).

When the autocommit mode is on, every query is automaticaly commited by the server, as if **ingres_commit()** was called after every call to **ingres_query()**.

See also **ingres_query()**, **ingres_rollback()** and **ingres_commit()**.

ingres_close

(PHP 4 >= 4.0.2, PHP 5)

ingres_close -- Close an Ingres II database connection

Description

bool **ingres_close** ([resource link])

Returns **TRUE** on success, or **FALSE** on failure.

ingres_close() closes the connection to the Ingres server that's associated with the specified link. If the *link* parameter isn't specified, the last opened link is used.

ingres_close() isn't usually necessary, as it won't close persistent connections and all non-persistent connections are automatically closed at the end of the script.

See also **ingres_connect()**, and **ingres_pconnect()**.

ingres_commit

(PHP 4 >= 4.0.2, PHP 5)

ingres_commit -- Commit a transaction.

Description

bool **ingres_commit** ([resource link])

ingres_commit() commits the currently open transaction, making all changes made to the database permanent.

This closes the transaction. A new one can be open by sending a query with **ingres_query()**.

You can also have the server commit automaticaly after every query by calling **ingres_autocommit()** before opening the transaction.

See also **ingres_query()**, **ingres_rollback()** and **ingres_autocommit()**.

ingres_connect

(PHP 4 >= 4.0.2, PHP 5)

ingres_connect -- Open a connection to an Ingres II database.

Description

resource **ingres_connect** ([string database [, string username [, string password]]])

Returns a Ingres II link resource on success, or **FALSE** on failure.

ingres_connect() opens a connection with the Ingres database designated by *database*, which follows the syntax *[node_id::]dbname[/svr_class]*.

If some parameters are missing, **ingres_connect()** uses the values in `php.ini` for *ingres.default_database*, *ingres.default_user* and *ingres.default_password*.

The connection is closed when the script ends or when **ingres_close()** is called on this link.

All the other ingres functions use the last opened link as a default, so you need to store the returned value only if you use more than one link at a time.

Ejemplo 1. ingres_connect() example

```
<?php
    $link = ingres_connect ("mydb", "user", "pass")
        or die ("Could not connect");
    print ("Connected successfully");
    ingres_close ($link);
?>
```

Ejemplo 2. ingres_connect() example using default link

```
<?php
    ingres_connect ("mydb", "user", "pass")
        or die ("Could not connect");
    print ("Connected successfully");
    ingres_close ();
?>
```

See also **ingres_pconnect()**, and **ingres_close()**.

ingres_fetch_array

(PHP 4 >= 4.0.2, PHP 5)

ingres_fetch_array -- Fetch a row of result into an array.

Description

array **ingres_fetch_array** ([int result_type [, resource link]])

ingres_fetch_array() Returns an array that corresponds to the fetched row, or **FALSE** if there are no more rows.

This function is an extended version of **ingres_fetch_row()**. In addition to storing the data in the numeric indices of the result array, it also stores the data in associative indices, using the field names as keys.

If two or more columns of the result have the same field names, the last column will take precedence. To access the other column(s) of the same name, you must use the numeric index of the column or make an alias for the column.

```
ingres_query(select t1.f1 as foo t2.f1 as bar from t1, t2);
$result = ingres_fetch_array();
$foo = $result["foo"];
$bar = $result["bar"];
```

result_type can be II_NUM for enumerated array, II_ASSOC for associative array, or II_BOTH (default).

Speed-wise, the function is identical to **ingres_fetch_object()**, and almost as quick as **ingres_fetch_row()** (the difference is insignificant).

Ejemplo 1. ingres_fetch_array() example

```
<?php
ingres_connect ($database, $user, $password);

ingres_query ("select * from table");
while ($row = ingres_fetch_array()) {
    echo $row["user_id"];  # using associative array
    echo $row["fullname"];
    echo $row[1];          # using enumerated array
    echo $row[2];
}
?>
```

See also **ingres_query()**, **ingres_num_fields()**, **ingres_field_name()**, **ingres_fetch_object()** and **ingres_fetch_row()**.

ingres_fetch_object

(PHP 4 >= 4.0.2, PHP 5)

ingres_fetch_object -- Fetch a row of result into an object.

Description

object **ingres_fetch_object** ([int result_type [, resource link]])

ingres_fetch_object() Returns an object that corresponds to the fetched row, or **FALSE** if there are no more rows.

This function is similar to **ingres_fetch_array()**, with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

The optional argument *result_type* is a constant and can take the following values: II_ASSOC, II_NUM, and II_BOTH.

Speed-wise, the function is identical to **ingres_fetch_array()**, and almost as quick as **ingres_fetch_row()** (the difference is insignificant).

Ejemplo 1. ingres_fetch_object() example

```
<?php
ingres_connect ($database, $user, $password);
ingres_query ("select * from table");
while ($row = ingres_fetch_object()) {
    echo $row->user_id;
    echo $row->fullname;
}
?>
```

See also **ingres_query()**, **ingres_num_fields()**, **ingres_field_name()**, **ingres_fetch_array()** and **ingres_fetch_row()**.

ingres_fetch_row

(PHP 4 >= 4.0.2, PHP 5)

ingres_fetch_row -- Fetch a row of result into an enumerated array.

Description

array **ingres_fetch_row** ([resource link])

ingres_fetch_row() returns an array that corresponds to the fetched row, or **FALSE** if there are no more rows. Each result column is stored in an array offset, starting at offset 1.

Subsequent call to **ingres_fetch_row()** would return the next row in the result set, or **FALSE** if there are no more rows.

Ejemplo 1. ingres_fetch_row() example

```
<?php
ingres_connect ($database, $user, $password);

ingres_query ("select * from table");
while ($row = ingres_fetch_row()) {
    echo $row[1];
    echo $row[2];
}
?>
```

See also **ingres_num_fields()**, **ingres_query()**, **ingres_fetch_array()** and **ingres_fetch_object()**.

ingres_field_length

(PHP 4 >= 4.0.2, PHP 5)

ingres_field_length -- Get the length of a field.

Description

int **ingres_field_length** (int index [, resource link])

ingres_field_length() returns the length of a field. This is the number of bytes used by the server to store the field. For detailed information, see the Ingres/OpenAPI User Guide - Appendix C.

index is the number of the field and must be between 1 and the value given by **ingres_num_fields()**.

See also **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()** and **ingres_fetch_row()**.

ingres_field_name

(PHP 4 >= 4.0.2, PHP 5)

ingres_field_name -- Get the name of a field in a query result.

Description

string **ingres_field_name** (int index [, resource link])

ingres_field_name() returns the name of a field in a query result, or **FALSE** on failure.

index is the number of the field and must be between 1 and the value given by **ingres_num_fields()**.

See also **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()** and **ingres_fetch_row()**.

ingres_field_nullable

(PHP 4 >= 4.0.2, PHP 5)

ingres_field_nullable -- Test if a field is nullable.

Description

bool **ingres_field_nullable** (int index [, resource link])

ingres_field_nullable() returns **TRUE** if the field can be set to the **NULL** value and **FALSE** if it can't.

index is the number of the field and must be between 1 and the value given by **ingres_num_fields()**.

See also **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()** and **ingres_fetch_row()**.

ingres_field_precision

(PHP 4 >= 4.0.2, PHP 5)

ingres_field_precision -- Get the precision of a field.

Description

int **ingres_field_precision** (int index [, resource link])

ingres_field_precision() returns the precision of a field. This value is used only for decimal, float and money SQL data types. For detailed information, see the Ingres/OpenAPI User Guide - Appendix C.

index is the number of the field and must be between 1 and the value given by **ingres_num_fields()**.

See also **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()** and **ingres_fetch_row()**.

ingres_field_scale

(PHP 4 >= 4.0.2, PHP 5)

ingres_field_scale -- Get the scale of a field.

Description

int **ingres_field_scale** (int index [, resource link])

ingres_field_scale() returns the scale of a field. This value is used only for the decimal SQL data type. For detailed information, see the Ingres/OpenAPI User Guide - Appendix C.

index is the number of the field and must be between 1 and the value given by **ingres_num_fields()**.

See also **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()** and **ingres_fetch_row()**.

ingres_field_type

(PHP 4 >= 4.0.2, PHP 5)

ingres_field_type -- Get the type of a field in a query result.

Description

string **ingres_field_type** (int index [, resource link])

ingres_field_type() returns the type of a field in a query result, or **`FALSE`** on failure. Examples of types returned are "IIAPI_BYTE_TYPE", "IIAPI_CHA_TYPE", "IIAPI_DTE_TYPE", "IIAPI_FLT_TYPE", "IIAPI_INT_TYPE", "IIAPI_VCH_TYPE". Some of these types can map to more than one SQL type depending on the length of the field (see **ingres_field_length()**). For example "IIAPI_FLT_TYPE" can be a float4 or a float8. For detailed information, see the Ingres/OpenAPI User Guide - Appendix C.

index is the number of the field and must be between 1 and the value given by **ingres_num_fields()**.

See also **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()** and **ingres_fetch_row()**.

ingres_num_fields

(PHP 4 >= 4.0.2, PHP 5)

ingres_num_fields -- Get the number of fields returned by the last query

Description

int **ingres_num_fields** ([resource link])

ingres_num_fields() returns the number of fields in the results returned by the Ingres server after a call to **ingres_query()**

See also **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()** and **ingres_fetch_row()**.

ingres_num_rows

(PHP 4 >= 4.0.2, PHP 5)

ingres_num_rows -- Get the number of rows affected or returned by the last query

Description

int **ingres_num_rows** ([resource link])

For delete, insert or update queries, **ingres_num_rows()** returns the number of rows affected by the query. For other queries, **ingres_num_rows()** returns the number of rows in the query's result.

> **Nota:** This function is mainly meant to get the number of rows modified in the database. If this function is called before using **ingres_fetch_array()**, **ingres_fetch_object()** or **ingres_fetch_row()** the server will delete the result's data and the script won't be able to get them.
>
> You should instead retrieve the result's data using one of these fetch functions in a loop until it returns **FALSE**, indicating that no more results are available.

See also **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()** and **ingres_fetch_row()**.

ingres_pconnect

(PHP 4 >= 4.0.2, PHP 5)

ingres_pconnect -- Open a persistent connection to an Ingres II database.

Description

resource **ingres_pconnect** ([string database [, string username [, string password]]])

Returns a Ingres II link resource on success, or **FALSE** on failure.

See **ingres_connect()** for parameters details and examples. There are only 2 differences between **ingres_pconnect()** and **ingres_connect()** : First, when connecting, the function will first try to find a (persistent) link that's already opened with the same parameters. If one is found, an identifier for it will be returned instead of opening a new connection. Second, the connection to the Ingres server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (**ingres_close()** will not close links established by **ingres_pconnect()**). This type of link is therefore called 'persistent'.

See also **ingres_connect()**, and **ingres_close()**.

ingres_query

(PHP 4 >= 4.0.2, PHP 5)

ingres_query -- Send a SQL query to Ingres II

Description

bool **ingres_query** (string query [, resource link])

Returns **TRUE** on success, or **FALSE** on failure.

ingres_query() sends the given *query* to the Ingres server. This query must be a valid SQL query (see the Ingres SQL reference guide)

The query becomes part of the currently open transaction. If there is no open transaction, **ingres_query()** opens a new transaction. To close the transaction, you can either call **ingres_commit()** to commit the changes made to the database or **ingres_rollback()** to cancel these changes. When the script ends, any open transaction is rolled back (by calling **ingres_rollback()**). You can also use **ingres_autocommit()** before opening a new transaction to have every SQL query immediatly commited.

Some types of SQL queries can't be sent with this function :

- close (see **ingres_close()**).
- commit (see **ingres_commit()**).
- connect (see **ingres_connect()**).
- disconnect (see **ingres_close()**).
- get dbevent
- prepare to commit
- rollback (see **ingres_rollback()**).
- savepoint
- set autocommit (see **ingres_autocommit()**).
- all cursor related queries are unsupported

Ejemplo 1. ingres_query() example

```
<?php
ingres_connect ($database, $user, $password);

ingres_query ("select * from table");
while ($row = ingres_fetch_row()) {
    echo $row[1];
    echo $row[2];
}
?>
```

See also **ingres_fetch_array()**, **ingres_fetch_object()**, **ingres_fetch_row()**, **ingres_commit()**, **ingres_rollback()** and **ingres_autocommit()**.

ingres_rollback

(PHP 4 >= 4.0.2, PHP 5)

ingres_rollback -- Roll back a transaction.

Description

bool **ingres_rollback** ([resource link])

ingres_rollback() rolls back the currently open transaction, actualy canceling all changes made to the database during the transaction.

This closes the transaction. A new one can be open by sending a query with **ingres_query()**.

See also **ingres_query()**, **ingres_commit()** and **ingres_autocommit()**.

LVI. IRC Gateway Functions

Introducción

With IRCG you can rapidly stream XML data to thousands of concurrently connected users. This can be used to build powerful, extensible interactive platforms such as online games and webchats. IRCG also features support for a non-streaming mode where a helper application reformats incoming data and supplies static file snippets in special formats such as cHTML (i-mode) or WML (WAP). These static files are then delivered by the high-performance web server.

Up to v4, IRCG runs under these platforms:

- AIX
- FreeBSD
- HP-UX
- Irix
- Linux
- Solaris
- Tru64
- Windows

Instalación

Detailed installation instructions can be found at http://www.schumann.cx/ircg/. We urge you to use the provided installation script.

It is not recommended, but you can try enable IRCG support yourself. Provide the path to the ircg-config script, *--with-ircg-config=path/to/irc-config* and in addition add *--with-ircg* to your configure line.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

Constantes predefinidas

Esta extensión no tiene ninguna constante definida.

Tabla de contenidos

ircg_channel_mode

(PHP 4 >= 4.0.5, PHP 5)

ircg_channel_mode -- Set channel mode flags for user

Description

bool **ircg_channel_mode** (resource connection, string channel, string mode_spec, string nick)

Set channel mode flags for *channel* on server connected to by *connection*. Mode flags are passed in *mode_spec* and are applied to the user specified by *nick*.

Mode flags are set or cleared by specifying a mode character and prepending it with a plus or minus character, respectively. E.g. operator mode is granted by '+o' and revoked by '-o', as passed as *mode_spec*.

ircg_disconnect

(PHP 4 >= 4.0.4, PHP 5)

ircg_disconnect -- Close connection to server

Description

bool **ircg_disconnect** (resource connection, string reason)

ircg_disconnect() will close a *connection* to a server previously established with **ircg_pconnect()**.

See also: **ircg_pconnect()**.

ircg_eval_ecmascript_params

(PHP 4 >= 4.3.0, PHP 5)

ircg_eval_ecmascript_params -- Decodes a list of JS-encoded parameters

Description

array **ircg_eval_ecmascript_params** (string params)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

```
From the ircg_eval_ecmascript_params source file:
   /*
    * State 0: Looking for ' or digit
    * State 1: Assembling parameter inside '..'
    * State 2: After escape sign: Copies single char verbatim, go to 1
    * State 3: Assembling numeric para, no quotation
    * State 4: Looking for ",", skipping whitespace
    */
```

See also **ircg_lookup_format_messages()**.

ircg_fetch_error_msg

(PHP 4 >= 4.1.0, PHP 5)

ircg_fetch_error_msg -- Returns the error from previous IRCG operation

Description

array **ircg_fetch_error_msg** (resource connection)

ircg_fetch_error_msg() returns the error from a failed connection.

> **Nota:** Error code is stored in first array element, error text in second. The error code is ñalent to IRC reply codes as defined by RFC 2812.

Ejemplo 1. ircg_fetch_error_msg() example

```
<?php
if (!ircg_join ($id, "#php")) {
    $error = ircg_fetch_error_msg($id);
    echo "Can't join channel #php. Error code:
          $error[0] Description: $error[1]";
}
?>
```

ircg_get_username

(PHP 4 >= 4.1.0, PHP 5)

ircg_get_username -- Get username for connection

Description

string **ircg_get_username** (resource connection)

Function **ircg_get_username()** returns the username for the specified connection *connection*. Returns **`FALSE`** if *connection* died or is not valid.

ircg_html_encode

(PHP 4 >= 4.0.5, PHP 5)

ircg_html_encode -- Encodes HTML preserving output

Description

bool **ircg_html_encode** (string html_string [, bool auto_links [, bool conv_br]])

Encodes a HTML string *html_string* for output. This exposes the interface which the IRCG extension uses internally to reformat data coming from an IRC link. The function causes IRC color/font codes to be encoded in HTML and escapes certain entities.

ircg_ignore_add

(PHP 4 >= 4.0.5, PHP 5)

ircg_ignore_add -- Add a user to your ignore list on a server

Description

bool **ircg_ignore_add** (resource connection, string nick)

This function adds user *nick* to the ignore list of connection *connection*. Afterwards, IRCG will suppress all messages from this user through the associated connection.

See also: **ircg_ignore_del()**.

ircg_ignore_del

(PHP 4 >= 4.0.5, PHP 5)

ircg_ignore_del -- Remove a user from your ignore list on a server

Description

bool **ircg_ignore_del** (resource connection, string nick)

This function removes user *nick* from the IRCG ignore list associated with *connection*.

See also: **ircg_ignore_add()**.

ircg_invite

(PHP 4 >= 4.3.3, PHP 5)

ircg_invite -- Invites nickname to channel

Description

bool **ircg_invite** (resource connection, string channel, string nickname)

ircg_invite() will send an invitation to the user *nickname*, prompting him to join *channel*. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

ircg_is_conn_alive

(PHP 4 >= 4.0.5, PHP 5)

ircg_is_conn_alive -- Check connection status

Description

bool **ircg_is_conn_alive** (resource connection)

ircg_is_conn_alive() returns **TRUE** if *connection* is still alive and working or **FALSE**, if the connection has died for some reason.

ircg_join

(PHP 4 >= 4.0.4, PHP 5)

ircg_join -- Join a channel on a connected server

Description

bool **ircg_join** (resource connection, string channel [, string key])

Join the channel *channel* on the server connected to by *connection*. IRCG will optionally pass the room key *key*.

ircg_kick

(PHP 4 >= 4.0.5, PHP 5)

ircg_kick -- Kick a user out of a channel on server

Description

bool **ircg_kick** (resource connection, string channel, string nick, string reason)

Kick user *nick* from *channel* on server connected to by *connection*. *reason* should give a short message describing why this action was performed.

ircg_list

(PHP 4 >= 4.3.3, PHP 5)

ircg_list -- List topic/user count of channel(s)

Description

bool **ircg_list** (resource connection, string channel)

ircg_list() will request a list of users in the *channel*. The answer is sent to the output defined by ircg_set_file() or ircg_set_current(). Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplo 1. ircg_list() example

```
<?php

// connect to server
$id = ircg_pconnect($nickname, $ip, $port);

// set to output to a file
ircg_set_file($id, 'irc_output.html');

// try to join a channel
if (!ircg_join($id, $channel)) {
    echo "Cannot /join $channel<br />";
}

// send list command
ircg_list($id, $channel);

// wait for output to arrive
sleep(5);

// disconnect
ircg_disconnect($id,'Bye World');

// output everything
readfile('irc_output.html');

?>
```

This example will output something similar to:

```
...
Channel #channel has n users and the topic is 'Topic'
End of LIST
...
```

See also: **ircg_set_file()**, **ircg_set_current()**, and **ircg_who()**.

ircg_lookup_format_messages

(PHP 4 >= 4.0.5, PHP 5)

ircg_lookup_format_messages -- Check for the existence of a format message set

Description

bool **ircg_lookup_format_messages** (string name)

Check for the existence of the format message set *name*. Sets may be registered with **ircg_register_format_messages()**, a default set named *ircg* is always available. Returns **TRUE**, if the set exists and **FALSE** otherwise.

See also: **ircg_register_format_messages()**

ircg_lusers

(PHP 4 >= 4.3.3, PHP 5)

ircg_lusers -- IRC network statistics

Description

bool **ircg_lusers** (resource connection)

ircg_lusers() will request a statistical breakdown of users on the network connected to on *connection*. The answer is sent to the output defined by **ircg_set_file()** or **ircg_set_current()**. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

See also: **ircg_set_file()**, and **ircg_set_current()**.

ircg_msg

(PHP 4 >= 4.0.4, PHP 5)

ircg_msg -- Send message to channel or user on server

Description

bool **ircg_msg** (resource connection, string recipient, string message [, bool suppress])

ircg_msg() will send the message to a channel or user on the server connected to by *connection*. A *recipient* starting with # or & will send the *message* to a channel, anything else will be interpreted as a username.

Setting the optional parameter *suppress* to a **TRUE** value will suppress output of your message to your own *connection*. This so-called loopback is necessary, because the IRC server does not echo PRIVMSG commands back to us.

ircg_names

(PHP 4 >= 4.3.3, PHP 5)

ircg_names -- Query visible usernames

Description

bool **ircg_names** (int connection, string channel [, string target])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

See also **ircg_get_username()** and **ircg_lusers()**.

ircg_nick

(PHP 4 >= 4.0.5, PHP 5)

ircg_nick -- Change nickname on server

Description

bool **ircg_nick** (resource connection, string nick)

Change your nickname on the given *connection* to the one given in *nick*, if possible.

Will return **TRUE** on success and **FALSE** on failure.

ircg_nickname_escape

(PHP 4 >= 4.0.6, PHP 5)

ircg_nickname_escape -- Encode special characters in nickname to be IRC-compliant

Description

string **ircg_nickname_escape** (string nick)

Function **ircg_nickname_escape()** returns an encoded nickname specified by *nick* which is IRC-compliant.

See also: **ircg_nickname_unescape()**

ircg_nickname_unescape

(PHP 4 >= 4.0.6, PHP 5)

ircg_nickname_unescape -- Decodes encoded nickname

Description

string **ircg_nickname_unescape** (string nick)

Function **ircg_nickname_unescape()** returns a decoded nickname, which is specified in *nick*.

See also: **ircg_nickname_escape()**

ircg_notice

(PHP 4 >= 4.0.5, PHP 5)

ircg_notice -- Send a notice to a user on server

Description

bool **ircg_notice** (resource connection, string recipient, string message)

This function will send the *message* text to the user *nick* on the server connected to by *connection*. IRC servers and other software will not automatically generate replies to NOTICEs in contrast to

other message types.

ircg_oper

(PHP 4 >= 4.3.3, PHP 5)

ircg_oper -- Elevates privileges to IRC OPER

Description

bool **ircg_oper** (resource connection, string name, string password)

ircg_oper() will authenticate the logged in user on *connection* as an IRC operator. *name* and *password* must match a registered IRC operator account. Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

ircg_part

(PHP 4 >= 4.0.4, PHP 5)

ircg_part -- Leave a channel on server

Description

bool **ircg_part** (resource connection, string channel)

Leave the channel *channel* on the server connected to by *connection*.

ircg_pconnect

(PHP 4 >= 4.0.4, PHP 5)

ircg_pconnect -- Connect to an IRC server

Description

resource **ircg_pconnect** (string username [, string server_ip [, int server_port [, string msg_format [, array ctcp_messages [, array user_settings [, bool bailout_on_trivial]]]]]])

ircg_pconnect() will try to establish a connection to an IRC server and return a connection resource handle for further use.

The only mandatory parameter is *username*, this will set your initial nickname on the server. *server_ip* and *server_port* are optional and default to *127.0.0.1* and *6667*.

> **Nota:** For now parameter *server_ip* will not do any hostname lookups and will only accept IP addresses in numerical form. DNS lookups are expensive and should be done in the context of IRCG.

You can customize the output of IRC messages and events by selecting a format message set previously created with **ircg_register_format_messages()** by specifying the set's name in *msg_format*.

If you want to handle CTCP messages such as ACTION (/me), you need to define a mapping from CTCP type (e.g. ACTION) to a custom format string. Do this by passing an associative array as *ctcp_messages*. The keys of the array are the CTCP type and the respective value is the format message.

You can define "ident", "password", and "realname" tokens which are sent to the IRC server by setting these in an associative array. Pass that array as *user_settings*.

See also: **ircg_disconnect()**, **ircg_is_conn_alive()**, **ircg_register_format_messages()**.

ircg_register_format_messages

(PHP 4 >= 4.0.5, PHP 5)

ircg_register_format_messages -- Register a format message set

Description

bool **ircg_register_format_messages** (string name, array messages)

With **ircg_register_format_messages()** you can customize the way your IRC output looks like or which script functions are invoked on the client side.

- Plain channel message
- Private message received
- Private message sent
- Some user leaves channel
- Some user enters channel
- Some user was kicked from the channel
- Topic has been changed
- Error
- Fatal error
- Join list end(?)
- Self part(?)
- Some user changes his nick
- Some user quits his connection

- Mass join begin
- Mass join element
- Mass join end
- Whois user
- Whois server
- Whois idle
- Whois channel
- Whois end
- Voice status change on user
- Operator status change on user
- Banlist
- Banlist end
- %f - from
- %t - to
- %c - channel
- %r - plain message
- %m - encoded message
- %j - js encoded message
- 1 - mod encode
- 2 - nickname decode

See also: **ircg_lookup_format_messages()**.

ircg_set_current

(PHP 4 >= 4.0.4, PHP 5)

ircg_set_current -- Set current connection for output

Description

bool **ircg_set_current** (resource connection)

Select the current HTTP connection for output in this execution context. Every output sent from the server connected to by *connection* will be copied to standard output while using default formatting or a format message set specified by **ircg_register_format_messages()**.

See also: **ircg_register_format_messages()**.

ircg_set_file

(PHP 4 >= 4.2.0, PHP 5)

ircg_set_file -- Set logfile for connection

Description

bool **ircg_set_file** (resource connection, string path)

Function **ircg_set_file()** specifies a logfile *path* in which all output from connection *connection* will be logged. Returns **TRUE** on success, otherwise **FALSE**.

ircg_set_on_die

(PHP 4 >= 4.2.0, PHP 5)

ircg_set_on_die -- Set action to be executed when connection dies

Description

bool **ircg_set_on_die** (resource connection, string host, int port, string data)

In case of the termination of connection *connection* IRCG will connect to *host* at *port* (Note: host must be an IPv4 address, IRCG does not resolve host-names due to blocking issues), send *data* to the new host connection and will wait until the remote part closes connection. This can be used to trigger a PHP script for example.

This feature requires IRCG 3.

ircg_topic

(PHP 4 >= 4.0.5, PHP 5)

ircg_topic -- Set topic for channel on server

Description

bool **ircg_topic** (resource connection, string channel, string new_topic)

Change the topic for channel *channel* on the server connected to by *connection* to *new_topic*.

ircg_who

(PHP 4 >= 4.3.3, PHP 5)

ircg_who -- Queries server for WHO information

Description

bool **ircg_who** (resource connection, string mask [, bool ops_only])

ircg_who() will request a list of users whose nickname is matching *mask* on connected network *connection*. The optional parameter *ops_only* will shrink the list to server operators only.

The answer is sent to the output defined by **ircg_set_file()** or **ircg_set_current()**. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

See also: **ircg_set_file()**, and **ircg_set_current()**.

ircg_whois

(PHP 4 >= 4.0.5, PHP 5)

ircg_whois -- Query server for user information

Description

bool **ircg_whois** (resource connection, string nick)

Sends a query to the connected server *connection* to ask for information about the specified user *nick*.

LVII. Integración de Java y PHP

Introducción

Existen dos formas diferentes de integrar Java y PHP: en primer lugar, se puede integrar PHP dentro de un entorno de ejecución de servlets de Java, que en estos momentos es una solución mas estable y mas eficiente. La segunda opción es la de integrar Java dentro de PHP. La primera forma de integración se realiza a traves de un modulo SAPI que actua como interfaz del servidor de servlets. La segunda forma se realiza mediante esta extensión de Java.

Esta extensión de Java proporciona de forma sencilla los medios necesarios para crear e invocar métodos sobre objetos de Java desde PHP. La JVM se crea utilizando JNI y todo se ejecuta en un unico proceso.

Aviso

Esta extensión es *EXPERIMENTAL*. Esto significa que el comportamiento de esta extensión, los nombre de sus funciones y en definitiva TODO lo documentado sobre esta extensión, puede cambiar en una futura versión de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Requirimientos

Para utilizar esta extensión es necesario disponer de una máquina virtual java (JVM) instalada en el sistema.

Instalación

Para incluir el soporte de Java en PHP, es necesario añadir el parámetro *--with-java[=DIR]* a las opciones de configuración de PHP. Ademas, es necesario indicarle a traves de la opción DIR, el lugar donde se encuentra el directorio base de la instalación del JDK. Esta extensión solamente puede ser construida como un modulo compartido. En el archivo `php4/ext/java/README` se incluye mas información sobre como construir esta extensión.

> **Nota para los usuarios de Windows:** Para poder trabajar con este modulo en un entorno Windows con una versión de PHP <= 4.0.6, se debe copiar el archivo *jvm.dll* que se encuentra en el subdirectorio DLL del directorio PHP/Win32 de la distribución binaria de PHP, en el directorio SYSTEM32 de Windows (Normalmente es C:\WINNT\SYSTEM32 o C:\WINDOWS\SYSTEM32). Para versiones de PHP > 4.0.6 no es necesario realizar esta operación.

Configuración en tiempo de ejecución

El comportamiento de estas funciones está afectado por los valores definidos en `php.ini`.

Tabla 1. Opciones de configuració de Java

Nombre	Valor por defecto	Donde se cambia
java.class.path	NULL	PHP_INI_ALL
java.home	NULL	PHP_INI_ALL
java.library.path	NULL	PHP_INI_ALL
java.library	JAVALIB	PHP_INI_ALL

Para conocer todos los detalles y las definiciones de las constantes de PHP_INI_* se debe consultar la función **ini_set()**.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Esta extensión no tiene ninguna constante definida.

Ejemplos

Ejemplo 1. Ejemplo de Java

```
<?php
  // se obtiene la instancia de la clase de Java java.lang.System desde PHP
  $system = new Java('java.lang.System');

  // se muestran las propiedades del acceso que se realiza a Java
  print 'Version de Java='.$system->getProperty('java.version').' <br>';
  print 'Desarrollador de la JVM=' .$system->getProperty('java.vendor').' <br>';
  print 'Sistema Operativo='.$system->getProperty('os.name').' '.
              $system->getProperty('os.version').' on '.
              $system->getProperty('os.arch').' <br>';

  // ejemplo de java.util.Date
  $formatter = new Java('java.text.SimpleDateFormat',
                        "EEEE, MMMM dd, yyyy 'at' h:mm:ss a zzzz");

  print $formatter->format(new Java('java.util.Date'));
?>
```

Ejemplo 2. Ejemplo de AWT

```
<?php
  // Este ejemplo solo puede ser ejecutado como CGI.

  $frame  = new Java('java.awt.Frame', 'PHP');
  $button = new Java('java.awt.Button', 'Hola Mundo de Java!');

  $frame->add('North', $button);
  $frame->validate();
  $frame->pack();
  $frame->visible = True;

  $thread = new Java('java.lang.Thread');
  $thread->sleep(10000);

  $frame->dispose();
?>
```

Notas:

- *new Java()* crea una nueva instancia de una clase solamente si existe un constructor adecuado. Si no se le pasan parámetros y existe un constructor por defecto adecuado (como por ejemplo en el caso de *java.lang.System*) se podrá acceder a la mayor parte de sus metodos, ya que son estáticos.

- Al acceder a los miembros de una instancia, en primer lugar se buscarán las propiedades del bean y en segundo lugar los miembros publicos. En otras palabras, *print $date.time* se intentara resolver en primer lugar como *$date.getTime()* y posteriormente como *$date.time*.

- Tanto los miembros estaticos como los miembros de una instancia de un objeto pueden ser accedidos utilizando la misma sintaxis. Ademas, si el objeto es de tipo *java.lang.Class*, entonces se puede acceder a los miembros estaticos de la clase (tanto los atributos como los metodos).

- Las excepciones que se lanzan durante la ejecución se transforman en avisos de tipo "warning" de PHP y en resultados de tipo **NULL**. Los avisos de tipo "warning" se pueden eliminar añadiendo el prefijo "@" a la llamada del metodo. Las siguientes funciones de la API se pueden utilizar para obtener y borrar el ultimo error surgido

    - **java_last_exception_get()**

    - **java_last_exception_clear()**

- La resolución de la sobrecarga es uno de los problemas mas dificiles de resolver dadas las grandes diferencias entre PHP y Java en el tema del "tipado" de las variables. Esta extensión utiliza un metodo simple pero muy efectivo para determinar cual es la mejor decisión a tomar cuando se produce la sobrecarga.

  Ademas, los nombres de los metodos en PHP no distinguen entre mayusculas y minusculas, por lo que se aumenta la probabilidad de que se produzcan situaciones de sobrecarga.

  Una vez seleccionado el metodo, los parámetros se transforman si es necesario, incluso con la posibilidad de perder datos (por ejemplo, los numeros de tipo "double" se transforman en tipo boolean) (Nota del traductor: esta conversion parece totalmente absurda, asi que puede tratarse de un fallo de la documentacion de la version original).

- Las variables de tipo "array" y "hashtable" pueden ser utilizadas de forma indistinta. En PHP, las hashtables solo pueden incluir en las claves variables de tipo integer o string. Ademas, en Java los arrays cuyas variables son de algun tipo primitivo, no pueden contener huecos. Por ultimo, se debe recordar que este tipo de variables se pasan por valor, por lo que pueden llegar a consumir una cantidad apreciable de memoria y de tiempo.

SAPI de los servlets Java

Basandose en el mismo mecanismo que la extension de Java de PHP, la SAPI de los servlets Java permite ejecutar PHP como un servlet de Java. La ventaja mas significativa de esta forma de actuar es que se aprovechan las caracteristicas de "pooling" y de reutilizacion implementadas por la mayoria de servidores que permiten ejecutar servlets. El archivo `php4/sapi/README` contiene las instrucciones necesarias para compilar el modulo SAPI para los servlets Java. Notas:

- Aunque, en principio, este codigo podria ser ejecutado en cualquier servidor que permita la ejecucion de servlets, solo se ha probado en el servidor Jakarta/Tomcat desarrollado por la fundacion Apache. Por ese motivo, cualquier informacion sobre la forma de ejecutarlo en otros servidores, los errores encontrados, las soluciones planteadas, etc...sera una informacion muy apreciada y que animamos a que los desarrolladores envien a los responsables del desarrollo del proyecto PHP.

- Se pueden producir conflictos entre PHP y SAPI con respecto al directorio de trabajo. Mientras PHP se esta ejecutando, el servidor de servlets no podra cargar ninguna clase que se encuentre en el CLASSPATH si su ruta se especifica de forma relativa y tampoco podra encontrar el directorio de trabajo utilizado para las tareas de administracion y compilacion de JSP.

Tabla de contenidos

java_last_exception_clear

(PHP 4 >= 4.0.2)

java_last_exception_clear -- Borra la última excepción de Java

Descripción

void **java_last_exception_clear** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Ver la función **java_last_exception_get()** para obtener un ejemplo completo del uso de esta función.

java_last_exception_get

(PHP 4 >= 4.0.2)

java_last_exception_get -- Obtiene la última excepción de Java

Descripción

exception **java_last_exception_get** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

El siguiente ejemplo muestra el manejo de las excepciones de Java dentro de PHP:

Ejemplo 1. Manejando las excepciones de Java

```
<?php
  $stack = new Java('java.util.Stack');
  $stack->push(1);

  // Lo siguiente deberia tener exito
  $result = $stack->pop();
  $ex = java_last_exception_get();
  if (!$ex) print "$result\n";

  // Lo siguiente deberia fallar (se suprime el error con el simbolo @)
  $result = @$stack->pop();
  $ex = java_last_exception_get();
  if ($ex) print $ex->toString();

  // Borrado de la ultima excepcion
  java_last_exception_clear();
?>
```

LVIII. Funciones LDAP

Introducción

LDAP es el protocolo de acceso a directorios ligero (Lightweight Directory Access Protocol), un protocolo usado para acceder a "Servidores de Directorio". El directorio es una clase especial de base de datos que contiene información estructurada en forma de árbol.

El concepto es similar a la estructura de directorios de los discos duros, pero en este caso, el directorio raiz es "El Mundo" y los subdirectorios de primer nivel son los "países". Niveles inferiores de la estructura de directorio contienen entradas para compañías, organizaciones o lugares, y en niveles aún inferiores se encuentran las entradas para la gente, y quizás de equipos informáticos y documentos.

Para referirse a un archivo en un subdirectorio del disco duro se usa algo como

```
/usr/local/misapps/docs
```

Las barras marcan cada división en la referencia al archivo, y la secuencia es leida de izquierda a derecha.

El ñalente a la referencia a un archivo en LDAP es el "distinguished name" (nombre distinguible), abreviado como "db". Un ejemplo de dn podría ser.

```
cn=Pedro Pérez,ou=Contabilidad,o=Mi Compañía,c=ES
```

Las comas marcan cada división en la referencia, y la secuencia se lee de derecha a izquierda. Este dn se leería como:

```
country = ES
organization = Mi Compañía
organizationalUnit = Contabilidad
commonName = Pedro Pérez
```

De la misma manera que no hay reglas estrictas sobre como organizar la estructura de directorios de un disco duro, un administrador de un servidor de directorio puede establecer cualquier estructura que sea útil para sus propósitos. Sin embargo hay algunos acuerdos tácitos que siempre deben seguirse. El mensaje es que no se puede escribir código para acceder un directorio si no se conoce algo de su estructura, igual que no se puede usar una base de datos sin algún conocimiento sobre lo que está disponible en ella.

Información sobre LDAP se puede encontrar en:

- Netscape
- OpenLDAP Project

- LDAP World

Netscape SDK tiene una Guia de programación muy buena en HTML.

Requirimientos

Se necesita obtener y compilar las bibliotecas LDAP cliente de la Universidad de Michigan ldap-3.3 package, Netscape Directory SDK 3.0 ó OpenLDAP si queremos que PHP soporte LDAP.

Instalación

LDAP support in PHP is not enabled by default. You will need to use the *--with-ldap[=DIR]* configuration option when compiling PHP to enable LDAP support. DIR is the LDAP base install directory. To enable SASL support, be sure *--with-ldap-sasl[=DIR]* is used, and that `sasl.h` exists on the system.

> **Note to Win32 Users:** In order to enable this module on a Windows environment, you must copy several files from the DLL folder of the PHP/Win32 binary package to the SYSTEM folder of your windows machine. (Ex: `C:\WINNT\SYSTEM32`, or `C:\WINDOWS\SYSTEM`). For PHP <= 4.2.0 copy `libsasl.dll`, for PHP >= 4.3.0 copy `libeay32.dll` and `ssleay32.dll` to your SYSTEM folder.

Configuración en tiempo de ejecución

El comportamiento de estas funciones está afectado por los valores definidos en `php.ini`.

Tabla 1. LDAP configuration options

Name	Default	Changeable
ldap.max_links	"-1"	PHP_INI_SYSTEM

For further details and definitions of the PHP_INI_* constants, see the Apéndice H.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

LDAP_DEREF_NEVER (integer)

LDAP_DEREF_SEARCHING (integer)

LDAP_DEREF_FINDING (integer)

LDAP_DEREF_ALWAYS (integer)

LDAP_OPT_DEREF (integer)

LDAP_OPT_SIZELIMIT (integer)

LDAP_OPT_TIMELIMIT (integer)

LDAP_OPT_PROTOCOL_VERSION (integer)

LDAP_OPT_ERROR_NUMBER (integer)

LDAP_OPT_REFERRALS (integer)

LDAP_OPT_RESTART (integer)

LDAP_OPT_HOST_NAME (integer)

LDAP_OPT_ERROR_STRING (integer)

LDAP_OPT_MATCHED_DN (integer)

LDAP_OPT_SERVER_CONTROLS (integer)

LDAP_OPT_CLIENT_CONTROLS (integer)

LDAP_OPT_DEBUG_LEVEL (integer)

GSLC_SSL_NO_AUTH (integer)

GSLC_SSL_ONEWAY_AUTH (integer)

GSLC_SSL_TWOWAY_AUTH (integer)

Ejemplos

Recuperar informacion para todas las entradas donde el apellido empiece por "P" de un servidor de directorio, mostrando un extracto con el nombre y dirección de correo electrónico.

Ejemplo 1. Ejemplo de búsqueda LDAP

```
<?php
// La secuencia b&aacute;sica para trabajar con LDAP es conectar, autentificarse,
// buscar, interpretar el resultado de la b&uacute;squeda y cerrar la conexi&oacute;n.

echo "&lt;h3>Prueba de consulta LDAP&lt;/h3>";
echo "Conectando ...";
$ds=ldap_connect("localhost");  // Debe ser un servidor LDAP v&aacute;lido!
echo "El resultado de la conexi&oacute;n es ".$ds."&lt;p>";

if ($ds) {
    echo "Autentificandose  ...";
    $r=ldap_bind($ds);     // Autentificaci&oacute;n an&oacute;nima, t&iacute;picamente
                           // acceso de lectura
    echo "El resultado de la autentificaci&oacute;n es ".$r."&lt;p>";

    echo "Buscando (sn=P*) ...";
    // Busqueda de entradas por apellidos
    $sr=ldap_search($ds,"o=Mi Compa&ntilde;ia, c=ES", "sn=P*");
    echo "El resultado de la b&uacute;squeda es ".$sr."&lt;p>";

    echo "El n&uacute;mero de entradas devueltas es ".ldap_count_entries($ds,$sr)."&lt;

    echo "Recuperando entradas ...&lt;p>";
    $info = ldap_get_entries($ds, $sr);
    echo "Devueltos datos de ".$info["count"]." entradas:&lt;p>";

    for ($i=0; $i<$info["count"]; $i++) {
        echo "dn es: ". $info[$i]["dn"] ."&lt;br>";
        echo "La primera entrada cn es: ". $info[$i]["cn"][0] ."&lt;br>";
        echo "La primera entrada email es: ". $info[$i]["mail"][0] ."&lt;p>";
    }

    echo "Cerrando conexi&oacute;n";
    ldap_close($ds);

} else {
    echo "&lt;h4>Ha sido imposible conectar al servidor LDAP&lt;/h4>";
}
?>
```

Usando las llamadas LDAP de PHP

Antes de usarse las llamadas LDAP se debe saber ..

- El nombre o dirección del servidor de directorio que se va a usar
- El "dn base" del servidor (la parte del directorio global contenida en ese servidor, que puede ser por ejemplo "o=Mi Compañia,c=ES")
- Si es necesaria contraseña para acceder al servidor (muchos servidores ofrecen acceso de lectura para usuarios anónimos pero requieren un password para cualquier otro acceso)

La secuencia típica de llamadas LDAP suele implementarse en aplicaciones que siguen el siguiente patrón:

```
ldap_connect()    // establecer la conexión con el servidor
  |
ldap_bind()       // login anónimo o autentificado
  |
Hacer búsquedas o actualizaciones en el directorio
y mostrar los resultados
```

```
  |
ldap_close()      // Cerrar la conexión
```

Tabla de contenidos

ldap_8859_to_t61

(PHP 4 >= 4.0.2, PHP 5)

ldap_8859_to_t61 -- Translate 8859 characters to t61 characters

Description

string **ldap_8859_to_t61** (string value)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ldap_add

(PHP 3, PHP 4 , PHP 5)

ldap_add -- Añade entradas a un directorio LDAP

Descripción

int **ldap_add** (int identificador_de_conexion, string dn, array entrada)

Devuelve verdadero cuando finaliza correctamente y falso se produce un error.

La función **ldap_add()** se usa para añadir entradas o registros a un directorio LDAP. El DN ("distinguished name", nombre distinguible, la referencia de cualquier entrada LDAP) es especificado por dn. El array entrada especifica la información que quiere añadirse. Los valores del array son indexados por sus propios atributos. En caso de valores múltiples para un mismo atributo, son indexados usando enteros empezando con 0.

```
entry["atributo1"] = valor
entry["atributo2"][0] = valor1
entry["atributo2"][1] = valor2
```

Ejemplo 1. Ejemplo completo con login atentificado

```
<?php
$ds=ldap_connect("localhost");  // Asumimos que el servidor LDAP está en el
                                // servidor local

if ($ds) {
    // autentificarse con el dn apropiado para tener permisos de modificación
    $r=ldap_bind($ds,"cn=root, o=Mi Compañia, c=ES", "secreto");

    // prepare data
    $info["cn"]="Pedro Pérez";
    $info["sn"]="Pedro";
    $info["mail"]="pedro.p@algun.sitio";
    $info["objectclass"]="persona";

    // add data to directory
    $r=ldap_add($ds, "cn=Pedro Pérez, o=Mi Compañia, c=ES", $info);

    ldap_close($ds);
} else {
    echo "Ha sido imposible conectar al servidor LDAP";
}
?>
```

ldap_bind

(PHP 3, PHP 4 , PHP 5)

ldap_bind -- Autentifica en un directorio LDAP

Descripción

int **ldap_bind** (int identificador_de_conexion [, string rdn_del_usuario [, string contraseña]])

Se conecta a un directorio LDAP con un RDN y su contraseña. Devuelve verdadero cuando finaliza correctamente y falso se produce un error.

ldap_bind() se conecta al directorio con un determinado usuario. rdn_de_usuario y contraseña son opcionales. Si no son especificados, se intenta el acceso anónimo.

ldap_close

(PHP 3, PHP 4 , PHP 5)

ldap_close -- Cierra una conexión a un servidor LDAP

Descripción

int **ldap_close** (int identificador_de_conexion)

Devuelve verdadero cuando finaliza correctamente y falso se produce un error.

ldap_close() cierra la conexión con el servidor LDAP asociada con el *identificador_de_conexion* especificado.

Esta llamada es idéntica internamente a **ldap_unbind()**. La API LDAP usa la llamada **ldap_unbind ()**, y por lo tanto quizás deba usar esta llamada en lugar de **ldap_close()**.

ldap_compare

(PHP 4 >= 4.0.2, PHP 5)

ldap_compare -- Compare value of attribute found in entry specified with DN

Description

bool **ldap_compare** (resource link_identifier, string dn, string attribute, string value)

Returns **`TRUE`** if *value* matches otherwise returns **`FALSE`**. Returns -1 on error.

ldap_compare() is used to compare *value* of *attribute* to value of same attribute in LDAP directory entry specified with *dn*.

The following example demonstrates how to check whether or not given password matches the one defined in DN specified entry.

Ejemplo 1. Complete example of password check

```
<?php

$ds=ldap_connect("localhost");  // assuming the LDAP server is on this host

if ($ds) {

    // bind
    if (ldap_bind($ds)) {

        // prepare data
        $dn = "cn=Matti Meikku, ou=My Unit, o=My Company, c=FI";
        $value = "secretpassword";
        $attr = "password";

        // compare value
        $r=ldap_compare($ds, $dn, $attr, $value);

        if ($r === -1) {
            echo "Error: " . ldap_error($ds);
        } elseif ($r === true) {
            echo "Password correct.";
        } elseif ($r === false) {
            echo "Wrong guess! Password incorrect.";
        }

    } else {
        echo "Unable to bind to LDAP server.";
    }

    ldap_close($ds);

} else {
    echo "Unable to connect to LDAP server.";
}
?>
```

Aviso
ldap_compare() can NOT be used to compare BINARY values!

Nota: This function was added in 4.0.2.

ldap_connect

(PHP 3, PHP 4 , PHP 5)

ldap_connect -- Conecta con un servidor LDAP

Descripción

int **ldap_connect** ([string nombre_host [, int puerto]])

Devuelve un identificador de conexión positivo en caso de éxito, ó falso si ocurre algún error.

ldap_connect() establece una conexión con el servidor LDAP especificado en *nombre_host* y *puerto*. Ambos argumentos son opcionales. Si no se especifican, el identificador de la conexión LDAP actualmente abierta es devuelto. Si sólo es especificado *nombre_host* el puerto tomado por defecto es el 389.

ldap_count_entries

(PHP 3, PHP 4 , PHP 5)

ldap_count_entries -- Cuenta el número de entradas de una búsqueda

Descripción

int **ldap_count_entries** (int identificador_de_conexion, int identificador_de_resultado)

Devuelve el número de entradas del resultado o falso si ha ocurrido algún error.

ldap_count_entries() devuelce el número de entradas almacenadas en el resultado de operaciones de búsqueda previas. *identificador_de_resultado* identifica el resultado ldap interno al que hacemos referencia.

ldap_delete

(PHP 3, PHP 4 , PHP 5)

ldap_delete -- Borra una entrada de un directorio

Descripción

int **ldap_delete** (int identificador_de_conexion, string dn)

Devuelve verdadero cuando finaliza correctamente y falso se produce un error.

La función **ldap_delete()** borra la entrada particular dn del directorio LDAP.

ldap_dn2ufn

(PHP 3, PHP 4 , PHP 5)

ldap_dn2ufn -- Convierte un dn al formato User Friendly Naming

Descripción

string **ldap_dn2ufn** (string dn)

La función **ldap_dn2ufn()** es usada para convertir un DN en un formato más amigable para el usuario.

ldap_err2str

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

ldap_err2str -- Convertir el número de error LDAP a un mensaje de error tipo cadena

Descripción

string **ldap_err2str** (int errno)

Devuelve el mensaje tipo cadena del error.

Esta función devuelve el mensaje de error explicando el número de error *errno*. Aunque los números errno de LDAP están estandarizados, bibliotecas diferentes devuelven mensajes de error textuales diferentes o incluso localizados. Nunca revise el texto de un mensaje de error específico, siempre use un número de error para efectos de chequeo.

Vea también **ldap_errno()** y **ldap_error()**.

Ejemplo 1. Enumeración de todos los mensajes de error LDAP

```
<?php
  for ($i=0; $i<100; $i++) {
    printf("Error $i: %s<br />\n", ldap_err2str($i));
  }
?>
```

ldap_errno

(PHP 3>= 3.0.12, PHP 4 , PHP 5)

ldap_errno -- Devuelve el número de error LDAP del último comando LDAP

Descripción

int **ldap_errno** (resource id_enlace)

Devuelve el número de error LDAP del último comando LDAP para este enlace.

Esta función devuelve el código numérico de error estándar devuelto por el último comando LDAP en el *id_enlace* dado. Este número puede ser convertido en un mensaje textual de error usando **ldap_err2str()**.

A menos que decremente lo suficiente su nivel de advertencia en `php.ini` o anteponga a los comandos LDAP el símbolo @ (arroba) para suprimir la salida de advertencias, los errores generados serán mostrados también en su salida HTML.

Ejemplo 1. Generación y captura de un error

```
<?php
// Este ejemplo contiene un error, el cual atraparemos.
$ld = ldap_connect("localhost");
$bind = ldap_bind($ld);

// error de sintaxis en la expresion de filtro (errno 87),
// debe ser "objectclass=*" para que funcione.
$res =  @ldap_search($ld, "o=Miorg, c=DE", "objectclass");

if (!$res) {
    echo "LDAP-Errno: " . ldap_errno($ld) . "<br />\n";
    echo "LDAP-Error: " . ldap_error($ld) . "<br />\n";
    die("&iexcl;Argh!<br />\n");
}

$info = ldap_get_entries($ld, $res);
echo $info["count"] . " entradas coincidentes.<br />\n";
?>
```

Vea también **ldap_err2str()** y **ldap_error()**.

ldap_error

(PHP 3>= 3.0.12, PHP 4 , PHP 5)

ldap_error -- Devuelve el mensaje de error LDAP del último comando LDAP

Descripción

string **ldap_error** (resource id_enlace)

Devuelve la cadena del mensaje de error.

Esta función devuelve la cadena del mensaje de error que explica el error generado por el último comando LDAP para el *id_enlace* dado. Aunque los números de error LDAP son estándar, diferentes bibliotecas devuelven mensajes de error textuales diferentes o incluso localizados. Nunca se debe verificar la existencia de un mensaje de texto de error específico, en su lugar use siempre un número de error para los chequeos.

A menos que decremente lo suficiente su nivel de advertencia en `php.ini` o anteponga a los comandos LDAP el símbolo @ (arroba) para suprimir la salida de advertencias, los errores generados serán mostrados también en su salida HTML.

Vea también **ldap_err2str()** y **ldap_errno()**.

ldap_explode_dn

(PHP 3, PHP 4 , PHP 5)

ldap_explode_dn -- Divide un DN en las partes que le componen

Descripción

array **ldap_explode_dn** (string dn, int con_atributos)

La función **ldap_explode_dn()** es usada para dividir un DN devuelto por **ldap_get_dn()** en las

partes que le componen. Cada parte es conocida como Relative Distinguished Name (Nombre Relativo Distinguible) abreviado como RDN. **ldap_explode_dn()** devuelve un array con todos esos componentes. *con_atributos* sirve para especificar si los RDN se devuelven sólo como valores o con sus atributos también (es decir, en un formato atributo=valor). Hay que poner *with_attrib* a 0 para obtener también los atributos y a 1 para obtener sólo los valores.

ldap_first_attribute

(PHP 3, PHP 4 , PHP 5)

ldap_first_attribute -- Devuelte el primer atributo

Descripción

string **ldap_first_attribute** (int identificador_de_conexion, int identificador_de_entrada_en_resultado, int identificador_ber)

Devuelte el primer atributo en la entrada o falso si ocurre algún error.

De manera similar a leer entradas, los atributos también son leidos de uno en uno de una entrada en particular del directorio. **ldap_first_attribute()** devuelve el primer atributo en la entrada a la que apunta el identificador_de_entrada_en_resultado El resto de los atributos son obtenidos llamanda a la función **ldap_next_attribute()** sucesivamente. El parámetro *indetificador_ber* es el identificador del puntero interno a memoria. Es pasado por referencia. El mismo *indetificador_ber* es pasado a la función **ldap_next_attribute()** que modifica dicho puntero.

Ver también **ldap_get_attributes()**

ldap_first_entry

(PHP 3, PHP 4 , PHP 5)

ldap_first_entry -- Devuelve el identificador del primer resultado

Descripción

int **ldap_first_entry** (int identificador_de_conexion, int identificador_de_resultado)

Devuelve el identificador de la primera entrada del resultado ó falso en caso de error.

Las entradas en un resultado LDAP son leidas secuencialmente usando las funciones **ldap_first_entry()** y **ldap_next_entry()**. **ldap_first_entry()** devuelve el identificador de la primera entrada del resultado. Este identificador es entonces suministrado a la rutina **lap_next_entry()** pata obtener sucesivas entradas del resultado.

Ver también **ldap_get_entries()**.

ldap_first_reference

(PHP 4 >= 4.0.5, PHP 5)

ldap_first_reference -- Return first reference

Description

resource **ldap_first_reference** (resource link, resource result)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ldap_free_result

(PHP 3, PHP 4 , PHP 5)

ldap_free_result -- Libera la memoria que almacena los resultados

Descripción

int **ldap_free_result** (int identificador_de_resultado)

Devuelve verdadero cuando finaliza correctamente y falso se produce un error.

ldap_free_result() libera la memoria reservada internamente para almacenar el resultado de búsquedas LDAP asociada al identificador *identificador_de_resultado*. Toda la memoria de resultados es automáticamente liberada al finalizarse la ejecución de un script.

Normalmente la memoria reservada para resultados ldap se libera al final del script. En caso de que el script realice sucesivas búsquedas que devuelvan conjuntos de resultados grandes, puede utilizarse **ldap_free_result()** para mantener bajo el uso de memoria del script durante su ejecución.

ldap_get_attributes

(PHP 3, PHP 4 , PHP 5)

ldap_get_attributes -- Obtiene los atributos de una entrada de un resultado de búsqueda

Descripción

array **ldap_get_attributes** (int identificador_de_conexion, int identificador_de_entrada_de_resultado)

Devuelve una completa información de la entrada en un array multidimensional o falso en caso de error.

La función **ldap_get_attributes()** es usada para simplificar el leer atributos y valores de una entrada de un resultado de búsqueda. El valor de retorno es un array multidimensional de atributos y sus valores.

Teniendo localizado una entrada específica en el directorio se puede conseguir la información que contiene dicha entrada usando esta llamada. Puede usar esta función para aplicaciones que naveguen

por las entradas del directorio y/o cuando no se conoce la estructura de las entradas del directorio. En otras aplicaciones se busca un atributo específico, como la dirección de email o los apellidos y no importa el resto de información contenida..

valor_devuelto["count"] = número de atributos en la entrada
valor_devuelto[0] = primer atributo
valor_devuelto[n] = enésimo atributo

valor_devuelto["atributo"]["count"] = número de vaslores del atributo
valor_devuelto["atributo"][0] = primer valor del atributo
valor_devuelto["atributo"][i] = iésimo valor del atributo

Ejemplo 1. Mostrar la lista de atributos contenida en una entrada específica de un directorio

```
// $ds es un identificador de conexión al directorio

// $sr es un resultado de búsqueda válido de una llamada
// anterior a una de las funciones de búsqueda en directorios
// ldap.

$entrada = ldap_first_entry($ds, $sr);

$atributos = ldap_get_attributes($ds, $entrada);

echo $atributos["count"]." atributos contenidos en esta entrada:<p>";

for ($i=0; $i < $atributos["count"]; $i++)
    echo $atributos[$i]."<br>";
```

Ver también **ldap_first_attribute()** y **ldap_next_attribute()**

ldap_get_dn

(PHP 3, PHP 4 , PHP 5)

ldap_get_dn -- Obtiene el DN de una entrada de un resultado

Descripción

string **ldap_get_dn** (int indentificador_de_conexion, int indentificador_de_entrada_de_resultado)

Devuelve el DN de la entrada del resultado o falso en caso de error.

La función **ldap_get_dn()** se utiliza para obtener el DN de una entrada de un resultado de búsqueda.

ldap_get_entries

(PHP 3, PHP 4 , PHP 5)

ldap_get_entries -- Obtiene todas las entradas de un resultado

Descripción

array **ldap_get_entries** (int indentificador_de_conexion, int indentificador_de_resultado)

Devuelve una completa información de un resultado de búsqueda en un array multidimensional o falso en caso de error.

La función **ldap_get_entries()** es usada para simplificar el leer múltiples entradas de de un resultado y después leer sus atributos y multiples valores. Toda la información es devuelta por una llamada a una función en forma de array multidimensional. La estructura del array es como se muestra más abajo.

Los indices de atributos son convertidos a minúsculas. (Los atributos de servidores de directorios son indiferentes a las mayusculas/minusculas, pero no cuando son usados como indices de arrays)

valor_devuelto["count"] = número de entradas del resultado
valor_devuelto[0] : contiene los detalles de la primera entrada

valor_devuelto[i]["dn"] = DN de la entrada iésima del resultado

valor_devuelto[i]["count"] = número de atributos de la entrada iésima
valor_devuelto[i][j] = jésimo atributo de la iésima entrada del resultado

valor_devuelto[i]["atributo"]["count"] = número de valores para "atributo"
en la entrada iésima
valor_devuelto[i]["atributo"][j] = jésimo valor de "atributo" en la entrada
iésima

Ver también **ldap_first_entry()** y **ldap_next_entry()**

ldap_get_option

(PHP 4 >= 4.0.4, PHP 5)

ldap_get_option -- Get the current value for given option

Description

bool **ldap_get_option** (resource link_identifier, int option, mixed &retval)

Sets *retval* to the value of the specified option. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

The parameter *option* can be one of: LDAP_OPT_DEREF, LDAP_OPT_SIZELIMIT, LDAP_OPT_TIMELIMIT, LDAP_OPT_PROTOCOL_VERSION, LDAP_OPT_ERROR_NUMBER, LDAP_OPT_REFERRALS, LDAP_OPT_RESTART, LDAP_OPT_HOST_NAME, LDAP_OPT_ERROR_STRING, LDAP_OPT_MATCHED_DN. These are described in draft-ietf-ldapext-ldap-c-api-xx.txt

> **Nota:** This function is only available when using OpenLDAP 2.x.x OR Netscape Directory SDK x.x, and was added in PHP 4.0.4

Ejemplo 1. Check protocol version

```
<?php
// $ds is a valid link identifier for a directory server
if (ldap_get_option($ds, LDAP_OPT_PROTOCOL_VERSION, $version)) {
    echo "Using protocol version $version\n";
} else {
    echo "Unable to determine protocol version\n";
}
?>
```

See also **ldap_set_option()**.

ldap_get_values_len

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

ldap_get_values_len -- Obtiene todos los valores binarios de un atributo de una entrada

Description

array **ldap_get_values_len** (int indentificador_de_conexion, int indentificador_de_entrada_de_resultado, string atributo)

Devuelve un array de valores del atributo o falso en caso de error.

La función **ldap_get_values_len()** se utiliza para obtener todos los valores de un atributo de una entrada de un resultado de búsqueda. La entrada es especificada por el *indentificador_de_entrada_de_resultado*. El número de valores se almacena en el índice "count" del array devuelto. Los valores individuales se almacenan con índices enteros en el array. El primer índice es 0.

Esta función se utiliza exactamente como **ldap_get_values()** salvo que permite manejar datos binarios y no cadenas de caracteres.

ldap_get_values

(PHP 3, PHP 4 , PHP 5)

ldap_get_values -- Obtiene todos los valores de un atributo de una entrada

Descripción

array **ldap_get_values** (int identificador_de_conexion, int identificador_de_entrada_de_resultado, string atributo)

Devuelve un array de valores del atributo o falso en caso de error.

La función **ldap_get_values()** se utiliza para obtener todos los valores de un atributo de una entrada. La entrada del resultado es especificada por el *identificador_de_entrada_de_resultado*. El número de valores se almacena en el índice "count" del array devuelto. Los valores individuales se almacenan con índices enteros en el array. El primer índice es 0.

Esta llamada necesita un *identificador_de_entrada_de_resultado*, por lo que necesita ser precedida por una de las llamadas de búsqueda ldap y una llamada para obtener una entrada en particular del

resultado.

La aplicación debe ser o bien programada específicamente para buscar ciertos atributos (como apellidos o email) o bien utilizar la función **ldap_get_attributes()** para averiguar que atributos existen para una entrada dada, antes de llamar a **ldap_get_values()**.

LDAP permite mas de un valor para cada atributo, por lo que se puede, por ejemplo, almacenar varias direcciones de email para una persona en el directorio y nombrar a ese atributo como "email"

valor_devuelto["count"] = número de valores del atributo
valor_devuelto[0] = primer valor del atributo
valor_devuelto[i] = iésimo valor del atributo

Ejemplo 1. Listar todos los valores del atributo "email" de una entrada de un directorio

```
// $ds es un identificador de conexión al directorio

// $sr es un resultado de búsqueda válido de una llamada
// anterior a una de las funciones de búsqueda en directorios
// ldap.

// $entrada es un identificador de entrada válido de una llamada
// anterior a una de las funciones que devuelven una entrada de
// directorio

$valores = ldap_get_values($ds, $entrada,"email");

echo $valores["count"]." direcciones de email para esta entrada.<p>";

for ($i=0; $i < $valores["count"]; $i++)
    echo $valores[$i]."<br>";
```

ldap_list

(PHP 3, PHP 4 , PHP 5)

ldap_list -- Búsqueda de nivel único

Descripción

resource **ldap_list** (resource id_enlace, string dn_base, string filtro [, array atributos [, int solo_atribs [, int limite_tamanyo [, int limite_tiempo [, int deref]]]]])

Devuelve un identificador de resultado de búsqueda, o **FALSE** en caso de error.

ldap_list() realiza la búsqueda de un *filtro* especificado en el directorio con el contexto LDAP_SCOPE_ONELEVEL.

LDAP_SCOPE_ONELEVEL quiere decir que la búsqueda sólo debe devolver información que se encuentre en el nivel inmediatamente inferior al *dn_base* especificado en la llamada. (ñalente a escribir "ls" y obtener un listado de archivos y carpetas en el directorio de trabajo actual.)

Esta llamada recibe 5 parámetros opcionales. Vea las notas de **ldap_search()**.

> **Nota:** Estos parámetros opcionales fueron agregados en 4.0.2: *solo_atribs*, *limite_tamanyo*, *limite_tiempo*, *deref*.

Ejemplo 1. Producir una lista de todas los unidades organizacionales de una organización

```
// $ds es un identificador de enlace valido para un servidor de directorios

$dn_base = "o=Mi Compa&ntilde;&iacute;a, c=ES";
$solo_estos = array("ou");

$sr=ldap_list($ds, $dn_base, "ou=*", $solo_estos);

$info = ldap_get_entries($ds, $sr);

for ($i=0; $i<$info["count"]; $i++) {
    echo $info[$i]["ou"][0] ;
}
```

> **Nota:** A partir de 4.0.5, es posible realizar búsquedas paralelas. Vea **ldap_search()** para más detalles.

ldap_mod_add

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

ldap_mod_add -- Añade valores de atributos

Descripción

int **ldap_mod_add** (int identificador_de_conexion, string dn, array entrada)

Devuelve verdadero cuando finaliza correctamente y falso se produce un error.

Esta función añadir uno o varios atributos al dn especificado. Realiza la modificación al nivel de atributos, en vez de hacerlo al nivel de objetos. Las modificaciones a nivel de objeto son propocionadas por la función **ldap_add()**.

ldap_mod_del

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

ldap_mod_del -- Borra valores de atributos

Descripción

int **ldap_mod_del** (int identificador_de_conexion, string dn, array entrada)

returns **TRUE** on success and **FALSE** on error.

Esta función elimina atributos del dn especificado. Realiza la modificación a nivel de atributos, en vez de hacerlo a nivel de objetos. Las modificaciones a nivel de objeto son proporcionadas por la función **ldap_del()**.

ldap_mod_replace

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

ldap_mod_replace -- Reemplaza valores de atributos

Descripción

int **ldap_mod_replace** (int identificador_de_conexion, string dn, array entrada)

Devuelve verdadero cuando finaliza correctamente y falso se produce un error.

Esta función reemplaza atributos del dn especificado. Realiza la modificación a nivel de atributos, en vez de hacerlo a nivel de objetos. Las modificaciones a nivel de objeto son proporcionadas por la función **ldap_modify()**.

ldap_modify

(PHP 3, PHP 4 , PHP 5)

ldap_modify -- Modifica una entrada LDAP

Descripción

int **ldap_modify** (int identificador_de_conexion, string dn, array entrada)

Devuelve verdadero cuando finaliza correctamente y falso se produce un error.

La función **ldap_modify()** se utiliza para modificar entradas existentes en un directorio LDAP. La estructura de la entrada es igual a la de **ldap_add()**.

ldap_next_attribute

(PHP 3, PHP 4 , PHP 5)

ldap_next_attribute -- Obtiene el siguiente atributo de una entrada

Descripción

string **ldap_next_attribute** (int identificador_de_conexion, int identificador_de_entrada_de_resultado, int identificador_ber)

Devuelve el siguiente atributo de una entrada o falso en caso de error.

ldap_next_attribute() es llamado para recuperar los atributos de una entrada. El estado interno del puntero es mantenido por el *identificador_ber*, que es pasado por referencia a la función. La primera llamada a **ldap_next_attribute()** es realizada con el *identificador_de_entrada_de_resultado* devuelto por la función **ldap_first_attribute()**.

Ver también **ldap_get_attributes()**

ldap_next_entry

(PHP 3, PHP 4 , PHP 5)

ldap_next_entry -- Obtiene la siguiente entrada de un resultado

Descripción

int **ldap_next_entry** (int identificador_de_conexion, int identificador_de_entrada_de_resultado)

Devuelve el identificador de la siguiente entrada del resultado. Las entradas deben haber sido leidas al principio con **ldap_first_entry()**. Si no hay más entradas en el resultado devuelve falso.

La función **ldap_next_entry()** se utiliza para obtener las entradas almacenadas en un resultado. Llamadas sucesivas a la función **ldap_next_entry()** devuelven las entradas una a una hasta que ya no queden más entradas. La primera llamada a **ldap_next_entry()** se realiza después de llamar a **ldap_first_entry()**.

Ver también **ldap_get_entries()**

ldap_next_reference

(PHP 4 >= 4.0.5, PHP 5)

ldap_next_reference -- Get next reference

Description

resource **ldap_next_reference** (resource link, resource entry)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ldap_parse_reference

(PHP 4 >= 4.0.5, PHP 5)

ldap_parse_reference -- Extract information from reference entry

Description

bool **ldap_parse_reference** (resource link, resource entry, array &referrals)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ldap_parse_result

(PHP 4 >= 4.0.5, PHP 5)

ldap_parse_result -- Extract information from result

Description

bool **ldap_parse_result** (resource link, resource result, int &errcode [, string &matcheddn [, string &errmsg [, array &referrals]]])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ldap_read

(PHP 3, PHP 4 , PHP 5)

ldap_read -- Lee una entrada

Descripción

int **ldap_read** (int identificador_de_conexión, string dn_base, string filtro [, array atributos])

Devuelve un identificador de resultado de búsqueda o falso en caso de error.

ldap_read() realiza la búsqueda según el filtro especificado con alcance LDAP_SCOPE_BASE, por lo que es ñalente a leer cualquier entrada del directorio.

No se permiten filtros vacios. Si se pretende obtener absolutamente toda la información, se debe usar un filtro del tipo "objectClass=*". Si conoce que tipos de entradas son usadas en el servidor de directorio es conveniente usar el filtro apropiado, como por ejemplo "objectClass=inetOrgPerson".

Esta llamada toma un cuarto parámetro opcional que es un array de los atributos requeridos. Consulte las notas de la función **ldap_search()**.

ldap_rename

(PHP 4 >= 4.0.5, PHP 5)

ldap_rename -- Modify the name of an entry

Description

bool **ldap_rename** (resource link_identifier, string dn, string newrdn, string newparent, bool deleteoldrdn)

The entry specified by *dn* is renamed/moved. The new RDN is specified by *newrdn* and the new

parent/superior entry is specified by *newparent*. If the parameter *deleteoldrdn* is **TRUE** the old RDN value(s) is removed, else the old RDN value(s) is retained as non-distinguished values of the entry. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

> **Nota:** This function currently only works with LDAPv3. You may have to use **ldap_set_option()** prior to binding to use LDAPv3. This function is only available when using OpenLDAP 2.x.x OR Netscape Directory SDK x.x, and was added in PHP 4.0.5.

ldap_sasl_bind

(PHP 5)

ldap_sasl_bind -- Bind to LDAP directory using SASL

Description

bool **ldap_sasl_bind** (resource link)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

> **Requirement: ldap_sasl_bind()** requires SASL support (`sasl.h`). Be sure *--with-ldap-sasl* is used when configuring PHP otherwise this function will be undefined.

ldap_search

(PHP 3, PHP 4 , PHP 5)

ldap_search -- Busca en un arbol LDAP

Descripción

int **ldap_search** (int identificador_de_conexion, string dn_base, string filtro [, array atributos])

Devuelve un identificador de resultado de búsqueda o falso en caso de error.

ldap_search() realiza la búsqueda según el filtro especificado con alcance LDAP_SCOPE_SUBTREE. Esto es ñalente a buscar en el directorio entero. *dn_base* especifica el DN base para el directorio.

Existe un cuarto parámetro opcional que puede ser añadido para restringir los atributos y valores devueltos por el servidor a sólo los requeridos. Es mucho más eficiente que la acción por defecto (que devolverá todos los atributos y sus valores asociados). El uso del cuarto parámetro debe ser por tanto considerado una práctica recomendable.

El cuerto parámetro es un array estándar de PHP con los atributos requeridos, por ejemplo array ("email","sn","cn"). Nota: "dn" siempre es devuelto independientemente de que tipos de atributos sean solicitados.

También es necesario resaltar que algunos servidores de directorio están configurados para devolver un cierto número de entradas como máximo. Si esto ocurre, el servidor indicará que solo devuelve un conjunto de resultados parcial.

El filtro de búsqueda puede ser simple o avanzado, usando operadores booleanos en el formato descrito en la documentación sobre LDAP (Consulte el Netscape Directory SDK para obtener completa información sobre filtros).

El ejemplo de abajo recupera la unidad organizativa (ou), apellidos nombre común y dirección de email para todas las personas de "Mi Compañía" donde los apellidos o el nombre común contiene la subcadena $persona. Este ejemplo usa un filtro booleano para indicar al servidor que busque la información en más de un atributo.

Ejemplo 1. Búsqueda LDAP

```
// $ds es un identificador de conexión válido

// $persona es todo o parte del nombre de una persona, por ejemplo "Pe"

$dn = "o=Mi Compañía, c=ES";
$filtro="(|(sn=$persona*)(givenname=$persona*))";
$solonecesito = array( "ou", "sn", "givenname", "mail");

$sr=ldap_search($ds, $dn, $filtro, $solonecesito);

$info = ldap_get_entries($ds, $sr);

print $info["count"]." entradas devueltas<p>";
```

ldap_set_option

(PHP 4 >= 4.0.4, PHP 5)

ldap_set_option -- Set the value of the given option

Description

bool **ldap_set_option** (resource link_identifier, int option, mixed newval)

Sets the value of the specified option to be *newval*. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo. on error.

The parameter *option* can be one of: LDAP_OPT_DEREF, LDAP_OPT_SIZELIMIT, LDAP_OPT_TIMELIMIT, LDAP_OPT_PROTOCOL_VERSION, LDAP_OPT_ERROR_NUMBER, LDAP_OPT_REFERRALS, LDAP_OPT_RESTART, LDAP_OPT_HOST_NAME, LDAP_OPT_ERROR_STRING, LDAP_OPT_MATCHED_DN, LDAP_OPT_SERVER_CONTROLS, LDAP_OPT_CLIENT_CONTROLS. Here's a brief description, see draft-ietf-ldapext-ldap-c-api-xx.txt for details.

The options LDAP_OPT_DEREF, LDAP_OPT_SIZELIMIT, LDAP_OPT_TIMELIMIT, LDAP_OPT_PROTOCOL_VERSION and LDAP_OPT_ERROR_NUMBER have integer value, LDAP_OPT_REFERRALS and LDAP_OPT_RESTART have boolean value, and the options LDAP_OPT_HOST_NAME, LDAP_OPT_ERROR_STRING and LDAP_OPT_MATCHED_DN have string value. The first example illustrates their use. The options LDAP_OPT_SERVER_CONTROLS and LDAP_OPT_CLIENT_CONTROLS require a list of controls, this means that the value must be an array of controls. A control consists of an *oid* identifying the control, an optional *value*, and an optional flag for *criticality*. In PHP a control is

given by an array containing an element with the key *oid* and string value, and two optional elements. The optional elements are key *value* with string value and key *iscritical* with boolean value. *iscritical* defaults to ***FALSE*** if not supplied. See also the second example below.

> **Nota:** This function is only available when using OpenLDAP 2.x.x OR Netscape Directory SDK x.x, and was added in PHP 4.0.4.

Ejemplo 1. Set protocol version

```
<?php
// $ds is a valid link identifier for a directory server
if (ldap_set_option($ds, LDAP_OPT_PROTOCOL_VERSION, 3)) {
    echo "Using LDAPv3";
} else {
    echo "Failed to set protocol version to 3";
}
?>
```

Ejemplo 2. Set server controls

```
<?php
// $ds is a valid link identifier for a directory server
// control with no value
$ctrl1 = array("oid" => "1.2.752.58.10.1", "iscritical" => true);
// iscritical defaults to FALSE
$ctrl2 = array("oid" => "1.2.752.58.1.10", "value" => "magic");
// try to set both controls
if (!ldap_set_option($ds, LDAP_OPT_SERVER_CONTROLS, array($ctrl1, $ctrl2)))
    echo "Failed to set server controls";
?>
```

See also **ldap_get_option()**.

ldap_set_rebind_proc

(PHP 4 >= 4.2.0, PHP 5)

ldap_set_rebind_proc -- Set a callback function to do re-binds on referral chasing

Description

bool **ldap_set_rebind_proc** (resource link, callback callback)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ldap_sort

(PHP 4 >= 4.2.0, PHP 5)

ldap_sort -- Sort LDAP result entries

Description

bool **ldap_sort** (resource link, resource result, string sortfilter)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ldap_start_tls

(PHP 4 >= 4.2.0, PHP 5)

ldap_start_tls -- Start TLS

Description

bool **ldap_start_tls** (resource link)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ldap_t61_to_8859

(PHP 4 >= 4.0.2, PHP 5)

ldap_t61_to_8859 -- Translate t61 characters to 8859 characters

Description

string **ldap_t61_to_8859** (string value)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ldap_unbind

(PHP 3, PHP 4 , PHP 5)

ldap_unbind -- Hace logout de un directorio LDAP

Descripción

int **ldap_unbind** (int identificador_de_conexion)

Devuelve verdadero cuando finaliza correctamente y falso se produce un error.

La función **ldap_unbind()** hace logout, desautentifica de un directorio LDAP.

LIX. libxml Functions

Introducción

These functions/constants are available since PHP 5.1.0 and if you have compiled one of the extensions based on libxml, like DOM, SimpleXML and XSLT.

Requirimientos

This extension requires libxml >= 2.6.0.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

`LIBXML_DTDATTR` (**integer**)

Default DTD attributes

`LIBXML_DTDLOAD` (**integer**)

Load the external subset

`LIBXML_DTDVALID` (**integer**)

Validate with the DTD

`LIBXML_ERR_ERROR` (**integer**)

A recoverable error

`LIBXML_ERR_FATAL` (**integer**)

A fatal error

`LIBXML_ERR_NONE` (**integer**)

No errors

`LIBXML_ERR_WARNING` (**integer**)

A simple warning

`LIBXML_NOBLANKS` (**integer**)

Remove blank nodes

LIBXML_NOCDATA (integer)

Merge CDATA as text nodes

LIBXML_NOENT (integer)

Substitute entities

LIBXML_NOERROR (integer)

Suppress error reports

LIBXML_NONET (integer)

Disable network access when loading documents

LIBXML_NOWARNING (integer)

Suppress warning reports

LIBXML_NSCLEAN (integer)

Remove redundant namespaces declarations

LIBXML_XINCLUDE (integer)

Implement XInclude substitution

Tabla de contenidos

libxml_clear_errors

(no version information, might be only in CVS)

libxml_clear_errors -- Clear libxml error buffer

Descripción

void **libxml_clear_errors** (void)

libxml_clear_errors() clears the libxml error buffer.

Valores retornados

No value is returned.

Ver también

libxml_get_errors()
libxml_get_last_error()

libxml_get_errors

(no version information, might be only in CVS)

libxml_get_errors -- Retrieve array of errors

Descripción

array **libxml_get_errors** (void)

Retrieve array of errors.

Valores retornados

Returns an array with LibXMLError objects if there are any errors in the buffer, or an empty array otherwise.

Ejemplos

Ejemplo 1. A libxml_get_errors() example

This example demonstrates how to build a simple libxml error handler.

```php
<?php

libxml_use_internal_errors(true);

$xmlstr = <<< XML
<?xml version='1.0' standalone='yes'?>
<movies>
 <movie>
  <titles>PHP: Behind the Parser</title>
 </movie>
</movies>
XML;

$doc = simplexml_load_string($xmlstr);

if (!$doc) {
    $errors = libxml_get_errors();

    foreach ($errors as $error) {
        echo display_xml_error($error);
    }

    libxml_clear_errors();
}


function display_xml_error($error) {

    switch ($error->level) {
        case LIBXML_ERR_WARNING:
            $return = "Warning $error->code: ";
            break;
         case LIBXML_ERR_ERROR:
            $return = "Error $error->code: ";
            break;
        case LIBXML_ERR_FATAL:
            $return = "Fatal Error $error->code: ";
            break;
    }

    $return .= trim($error->message) .
               "\n  Line: $error->line" .
               "\n  Column: $error->column";

    if ($error->file) {
        $return .= "\n  File: $error->file";
    }

    return "$return\n";
}

?>
```

El resultado del ejemplo seria:

```
Fatal Error 76: Opening and ending tag mismatch: titles line 4 and title
  Line: 4
  Column: 0
```

Ver también

libxml_get_last_error()
libxml_clear_errors()

libxml_get_last_error

(no version information, might be only in CVS)

libxml_get_last_error -- Retrieve last error from libxml

Descripción

LibXMLError **libxml_get_last_error** (void)

Retrieve last error from libxml.

Valores retornados

Returns a LibXMLError object if there is any error in the buffer, **FALSE** otherwise.

Ver también

libxml_get_errors()
libxml_clear_errors()

libxml_set_streams_context

(PHP 5)

libxml_set_streams_context -- Set the streams context for the next libxml document load or write

Descripción

void **libxml_set_streams_context** (resource streams_context)

Sets the streams context for the next libxml document load or write.

Lista de parámetros

streams_context

The stream context resource (created with **stream_context_create()**)

Valores retornados

No value is returned.

Ejemplos

Ejemplo 1. A libxml_set_streams_context() example

```php
<?php

$opts = array(
    'http' => array(
        'user_agent' => 'PHP libxml agent',
    )
);

$context = stream_context_create($opts);
libxml_set_streams_context($context);

// request a file through HTTP
$doc = DOMDocument::load('http://www.example.com/file.xml');

?>
```

Ver también

stream_context_create()

libxml_use_internal_errors

(no version information, might be only in CVS)

libxml_use_internal_errors -- Disable libxml errors and allow user to fetch error information as needed

Descripción

bool **libxml_use_internal_errors** ([bool use_errors])

libxml_use_internal_errors() allows you to disable standard libxml errors and enable user error handling.

Lista de parámetros

use_errors

Whether to enable user error handling. Defaults to **FALSE**.

Valores retornados

This function returns the previous value of *use_errors*.

Ejemplos

Ejemplo 1. A libxml_use_internal_errors() example

This example demonstrates the basic usage of libxml errors and the value returned by this function.

```php
<?php

// enable user error handling
var_dump(libxml_use_internal_errors(true));

$doc = DOMDocument::load('file.xml');

if (!$doc) {
    $errors = libxml_get_errors();
    foreach ($errors as $error) {
        // handle errors here
    }

    libxml_clear_errors();
}

?>
```

El resultado del ejemplo seria:

```
bool(false)
```

Ver también

libxml_clear_errors()
libxml_get_errors()

LX. LZF Functions

Introducción

LZF is a very fast compression algorithm, ideal for saving space with only slight speed cost. It can be optimized for speed or space at the time of compilation.

Instalación

Esta extension PECL no esta ligada a PHP. Mas informacion sobre nuevos lanzamientos, descargas ficheros de fuentes, informacion sobre los responsables asi como un 'CHANGELOG', se puede encontrar aqui: http://pecl.php.net/package/lzf.

In order to use these functions you must compile PHP with lzf support by using the *--with-lzf [=DIR]* configure option. You may also pass *--enable-lzf-better-compression* to optimize LZF for space rather then speed.

Windows users will enable `php_lzf.dll` inside of `php.ini` in order to use these functions. Podeis descargar esta DLL de las extensiones PECL desde la pagina PHP Downloads o desde http://snaps.php.net/.

Tabla de contenidos

lzf_compress

(no version information, might be only in CVS)

lzf_compress -- LZF compression

Descripción

string **lzf_compress** (string data)

lzf_compress() compresses the given *data*.

Lista de parámetros

data

The string to compress.

Valores retornados

Returns the compressed data or **FALSE** if an error occurred.

Ver también

lzf_decompress()

lzf_decompress

(no version information, might be only in CVS)

lzf_decompress -- LZF decompression

Descripción

string **lzf_decompress** (string data)

lzf_compress() decompresses the given *data*.

Lista de parámetros

data

The compressed string.

Valores retornados

Returns the decompressed data or **FALSE** if an error occurred.

Ver también

lzf_compress()

lzf_optimized_for

(no version information, might be only in CVS)

lzf_optimized_for -- Determines what LZF extension was optimized for

Descripción

int **lzf_optimized_for** (void)

Determines for what the LZF extension was optimised.

Valores retornados

Returns 1 if LZF was optimized for speed, 0 for compression.

LXI. Funciones de Correo

Introducción

La función **mail()** le permite enviar correo.

Requirimientos

Para que las funciones de Correo se encuentren disponibles, PHP debe tener acceso al binario *sendmail* en su sistema durante tiempo de compilación. Si usa otro programa de correo, como qmail o postfix, asegúrese de usar las envolturas sendmail apropiadas que vienen con tales sistemas de correo. PHP buscará sendmail primero en su *PATH*, y luego en los siguientes sitios: */usr/bin:/usr/sbin:/usr/etc:/etc:/usr/ucblib:/usr/lib*. Es bastante recomendable contar con el programa sendmail disponible en su *PATH*. Asimismo, el usuario que compile PHP debe tener permiso para acceder al binario sendmail.

Instalación

No se necesita ninguna instalación para usar estas funciones, son parte del núcleo de PHP.

Configuración en tiempo de ejecución

El comportamiento de estas funciones está afectado por los valores definidos en `php.ini`.

Tabla 1. Opciones de configuración de correo

Nombre	Predeterminado	Modificable
SMTP	"localhost"	PHP_INI_ALL
smtp_port	"25"	PHP_INI_ALL
sendmail_from	NULL	PHP_INI_ALL
sendmail_path	DEFAULT_SENDMAIL_PATH	PHP_INI_SYSTEM

Para más detalles sobre las constantes PHP_INI_* y su definición, vea **ini_set()**.

A continuación se presenta una corta explicación de las directivas de configuración.

SMTP **string**

Usado bajo Windows únicamente: el nombre DNS o dirección IP del servidor SMTP que debería usar PHP para el envío de correo con la función **mail()**.

smtp_port **int**

Usado bajo Windows únicamente: Número del puerto para conectarse al servidor especificado en el parámetro *SMTP* cuando se envíe correo con **mail()**; su valor predeterminado es 25. Se encuentra disponible solo a partir de PHP 4.3.0.

sendmail_from **string**

Qué dirección de correo "From:" debe ser usada en el correo enviado desde PHP bajo Windows.

sendmail_path **string**

En dónde puede encontrarse el programa **sendmail**, usualmente `/usr/sbin/sendmail` o `/usr/lib/sendmail`. **configure** realiza un honesto intento por ubicar este valor para usted y definir un valor predeterminado, pero si falla, puede definirlo aquí.

Los sistema que no usan sendmail deben definir esta directiva al reemplazo de sendmail que ofrecen sus sistemas de correo, si existe. Por ejemplo, los usuarios de Qmail pueden definir este valor normalmente a `/var/qmail/bin/sendmail` o `/var/qmail/bin/qmail-inject`.

qmail-inject no requiere ninguna opción para procesar el correo correctamente.

Esta directiva funciona también bajo Windows. Si está definida, *smtp*, *smtp_port* y *sendmail_from* son ignoradas y se ejecuta el comando especificado.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Esta extensión no tiene ninguna constante definida.

Tabla de contenidos

ezmlm_hash

(PHP 3>= 3.0.17, PHP 4 >= 4.0.2, PHP 5)

ezmlm_hash -- Calcular el valor hash que necesita EZMLM

Descripción

int **ezmlm_hash** (string dir)

ezmlm_hash() calcula el valor de hash que se necesita cuando se administran listas de correo EZMLM en una base de datos MySQL.

Ejemplo 1. Cálculo de hash y suscripción de un usuario

```
<?php

$usuario  = "joecool@example.com";
$hash     = ezmlm_hash($usuario);
$consulta = sprintf("INSERT INTO ejemplo VALUES (%s, '%s')", $hash, $usuario);
$db->query($consulta); // usando la interfaz de bd PHPLIB

?>
```

mail

(PHP 3, PHP 4 , PHP 5)

mail -- Enviar correo

Descripción

bool **mail** (string para, string asunto, string mensaje [, string cabeceras_adicionales [, string parametros_adicionales]])

mail() envía automáticamente por correo el mensaje especificado en *mensaje* al recipiente especificado en *para*. Es posible especificar múltiples recipientes colocando una coma entre cada dirección en la cadena *para*. Es posible enviar correo electrónico con archivos adjuntos y tipos especiales usando esta función. Esto se consigue mediante el uso de codificación MIME - para más información, vea este artículo de Zend o las Clases Mime PEAR.

Los siguientes RFC pueden resultar asimismo útiles: RFC 1896, RFC 2045, RFC 2046, RFC 2047, RFC 2048, y RFC 2049.

mail() devuelve **`TRUE`** si el correo fue aceptado satisfactoriamente para su envío, o **`FALSE`** de lo

contrario.

Aviso
La implementación de Windows de **mail()** difiere en varias formas de la implementación en Unix. Primero, no usa un binario local para componer mensajes, en su lugar opera directamente sobre sockets, lo que quiere decir que se requiere de un *MTA* que escuche sobre un socket de red (el cual puede estar en el host local o en una máquina remota). Segundo, las cabeceras personalizadas como *From:*, *Cc:*, *Bcc:* y *Date:* **no** son interpretadas por el *MTA* en primera instancia, son interpretadas por PHP. PHP < 4.3 sólo soportaba el elemento de cabecera *Cc:* (y era sensible a mayúsculas y minúsculas). PHP >= 4.3 soporta todos los elementos de cabecera mencionados y ya no es sensible a mayúsculas y minúsculas.

Ejemplo 1. Envío de correo

```
<?php
mail("juan-el-listo@example.com", "Mi Asunto", "Linea 1\nLinea 2\nLinea 3");
?>
```

Si un cuarto argumento tipo cadena es entregado, éste es insertado al final de la cabecera. Esto se usa típicamente con el propósito de agregar cabeceras adicionales. Las cabeceras extra son separadas entre sí con un retorno de carro y un salto de línea.

> **Nota:** Debe usar *\r\n* para separar las cabeceras, aun cuando puede que los agentes de transferencia de correo en Unix trabajen con solo una nueva línea (*\n*).

Ejemplo 2. Envío de correo con cabeceras adicionales

```
<?php
mail("nadie@example.com", "el asunto", $mensaje,
     "From: webmaster@{$_SERVER['SERVER_NAME']}\r\n" .
     "Reply-To: webmaster@{$_SERVER['SERVER_NAME']}\r\n" .
     "X-Mailer: PHP/" . phpversion());
?>
```

El valor *parametros_adicionales* puede ser usado para pasar un parámetro extra al programa configurado para ser usado cuando se envíe correo mediante la opción de configuración *sendmail_path*. Por ejemplo, éste parámetro puede usarse para definir la dirección de sobre del origen cuando se usa sendmail con la opción *-f*. Puede que también necesite agregar el usuario bajo el que corre su servidor web a la configuración de sendmail para evitar que se agregue una cabecera 'X-Warning' al mensaje cuando define el sobre del origen mediante este método.

Ejemplo 3. Envío de correo con cabeceras adicionales y definiendo un parámetro extra de línea de com

```
<?php
mail("nadie@example.com", "el asunto", $mensaje,
     "From: webmaster@{$_SERVER['SERVER_NAME']}", "-fwebmaster@{$_SERVER['SERVER_NAME']
?>
```

> **Nota:** Este quinto parámetro fue agregado en PHP 4.0.5. A partir de PHP 4.2.3 este parámetro se encuentra deshabilitado en safe_mode y la función **mail()** producirá un mensaje de advertencia y devolverá **FALSE** si trata de usarlo.

También puede usar técnicas simples de construcción de cadenas para generar mensajes complejos de correo electrónico.

Ejemplo 4. Envío de correo electrónico complejo

```
<?php
/* recipientes */
$para  = "maria@example.com" . ", " ; // fijese en la comma
$para .= "kelly@example.com";

/* asunto */
$asunto = "Recordatorio de Cumplea&ntilde;os para Agosto";

/* mensaje */
$mensaje = '
<html>
<head>
 <title>Recordatorio de Cumplea&ntilde;os para Agosto</title>
</head>
<body>
<p>&iexcl;Aqu&iacute; est&aacute;n los cumplea&ntilde;os que llegan en Agosto!</p>
<table>
 <tr>
  <th>Persona</th><th>D&iacute;a</th><th>Mes</th><th>A&ntilde;o</th>
 </tr>
 <tr>
  <td>Juan</td><td>3</td><td>August</td><td>1970</td>
 </tr>
 <tr>
  <td>Sandra</td><td>17</td><td>August</td><td>1973</td>
 </tr>
</table>
</body>
</html>
';

/* Para enviar correo HTML, puede definir la cabecera Content-type. */
$cabeceras  = "MIME-Version: 1.0\r\n";
$cabeceras .= "Content-type: text/html; charset=iso-8859-1\r\n";

/* cabeceras adicionales */
$cabeceras .= "To: Maria <maria@example.com>, Kelly <kelly@example.com>\r\n";
$cabeceras .= "From: Recordatorio <cumpleanyos@example.com>\r\n";
$cabeceras .= "Cc: archivo@example.com\r\n";
$cabeceras .= "Bcc: chequeo@example.com\r\n";

/* y ahora, enviarlo */
mail($para, $asunto, $mensaje, $cabeceras);
?>
```

Nota: Asegúrese de que no tiene caracteres de salto de línea en los parámetros *para* o *asunto*, o puede que el correo no sea enviado correctamente.

Nota: El parámetro *para* no debería ser una dirección en la forma de "Algo <alguien@example.com>". Es posible que el comando de correo no interprete esto correctamente mientras habla cen el MTA (particularmente bajo windows).

Vea también **imap_mail()**.

LXII. Funciones mailparse

Introducción

Aviso

Esta extensión es *EXPERIMENTAL*. Esto significa que el comportamiento de esta extensión, los nombre de sus funciones y en definitiva TODO lo documentado sobre esta extensión, puede cambiar en una futura versión de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Esta extensión ha sido movida desde PHP a partir de PHP 4.2.0 y ahora mailparse existe en PECL.

Instalación

Esta extension PECL no esta ligada a PHP. Mas informacion sobre nuevos lanzamientos, descargas ficheros de fuentes, informacion sobre los responsables asi como un 'CHANGELOG', se puede encontrar aqui: http://pecl.php.net/package/mailparse.

Para usar éstas funciones, es necesario compilar PHP con soporte mailparse mediante el uso de la opción de configuración *--enable-mailparse*.

Los usuarios de windows deben habilitar `php_mailparse.dll` al interior de `php.ini` para usar éstas funciones. Podeis descargar esta DLL de las extensiones PECL desde la pagina PHP Downloads o desde http://snaps.php.net/.

Tabla de contenidos

mailparse_determine_best_xfer_encoding

(4.1.0 - 4.1.2 only)

mailparse_determine_best_xfer_encoding -- Encuentra la mejor forma de codificar el contenido leído desde el apuntador de archivo fp, el cual debe tener capacidades de búsqueda

Descripción

int **mailparse_determine_best_xfer_encoding** (resource fp)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.
Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mailparse_msg_create

(4.1.0 - 4.1.2 only)

mailparse_msg_create -- Devuelve un gestor que puede ser usado para procesar un mensaje

Descripción

int **mailparse_msg_create** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.
Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mailparse_msg_extract_part_file

(4.1.0 - 4.1.2 only)

mailparse_msg_extract_part_file -- Extrae/decodifica una sección de mensaje, operando sobre la codificación de transferencia

Descripción

string **mailparse_msg_extract_part_file** (resource rfc2045, string nombre_archivo [, callback llamada_retorno])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.
Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mailparse_msg_extract_part

(4.1.0 - 4.1.2 only)

mailparse_msg_extract_part -- Extrae/decodifica una sección de mensaje

Descripción

void **mailparse_msg_extract_part** (resource rfc2045, string cuerpo_mensaje [, callback llamada_retorno])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

Si la llamada de retorno no se especifica, los contenidos serán enviados a "stdout".

mailparse_msg_free

(4.1.0 - 4.1.2 only)

mailparse_msg_free -- Libera un gestor ubicado por **mailparse_msg_create()**

Descripción

void **mailparse_msg_free** (resource rfc2045buf)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mailparse_msg_get_part_data

(4.1.0 - 4.1.2 only)

mailparse_msg_get_part_data -- Devuelve una matriz asociativa de información sobre el mensaje

Descripción

array **mailparse_msg_get_part_data** (resource rfc2045)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mailparse_msg_get_part

(4.1.0 - 4.1.2 only)

mailparse_msg_get_part -- Devuelve un gestor sobre una sección dada en un mensaje mime

Descripción

int **mailparse_msg_get_part** (resource rfc2045, string seccion_mime)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mailparse_msg_get_structure

(4.1.0 - 4.1.2 only)

mailparse_msg_get_structure -- Devuelve una matriz de nombres de sección mime en el mensaje dado

Descripción

array **mailparse_msg_get_structure** (resource rfc2045)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mailparse_msg_parse_file

(4.1.0 - 4.1.2 only)

mailparse_msg_parse_file -- Procesar un archivo y devolver un recurso que represente la estructura

Descripción

resource **mailparse_msg_parse_file** (string nombre_archivo)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mailparse_msg_parse

(4.1.0 - 4.1.2 only)

mailparse_msg_parse -- Procesar datos incrementalmente sobre un búfer

Descripción

void **mailparse_msg_parse** (resource rfc2045buf, string datos)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mailparse_rfc822_parse_addresses

(4.1.0 - 4.1.2 only)

mailparse_rfc822_parse_addresses -- Procesar direcciones y devolver una matriz asociativa que contenga los datos

Descripción

array **mailparse_rfc822_parse_addresses** (string direcciones)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mailparse_stream_encode

(4.1.0 - 4.1.2 only)

mailparse_stream_encode -- Secuencia datos desde un apuntador de archivo, codifica y escribe a fp_destino

Descripción

bool **mailparse_stream_encode** (resource fp_fuente, resource fp_destino, string codificacion)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mailparse_uudecode_all

(no version information, might be only in CVS)

mailparse_uudecode_all -- Procesa los datos desde un apuntador a archivo y extrae cada archivo embebido con codificación uu

Descripción

array **mailparse_uudecode_all** (resource fp)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

Devuelve una matriz que lista la información del archivo.

LXIII. Funciones matemáticas

Introducción

Estas funciones matemáticas solo manipulan valores dentro del rango de los tipos **integer** y **float** en su equipo (en la actualidad, estos tipos corresponden con los tipos long y double de C). Si necesita manejar números más grandes, eche un vistazo a las funciones matemáticas de precisión arbitraria.

Vea también la página del manual sobre operadores aritméticos.

Requirimientos

No se necesitan bibliotecas externas para construir esta extensión

---

Instalación

No se necesita ninguna instalación para usar estas funciones, son parte del núcleo de PHP.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Las constantes listadas aqui están siempre disponibles a través del "núcleo PHP".

Tabla 1. Constantes matemáticas

Constante	Valor	Descripción
M_PI	3.14159265358979323846	Pi
M_E	2.7182818284590452354	e
M_LOG2E	1.4426950408889634074	log_2 e
M_LOG10E	0.43429448190325182765	log_10 e
M_LN2	0.69314718055994530942	log_e 2
M_LN10	2.30258509299404568402	log_e 10
M_PI_2	1.57079632679489661923	pi/2
M_PI_4	0.78539816339744830962	pi/4
M_1_PI	0.31830988618379067154	1/pi
M_2_PI	0.63661977236758134308	2/pi
M_SQRTPI	1.77245385090551602729	sqrt(pi) [4.0.2]
M_2_SQRTPI	1.12837916709551257390	2/sqrt(pi)
M_SQRT2	1.41421356237309504880	sqrt(2)
M_SQRT3	1.73205080756887729352	sqrt(3) [4.0.2]
M_SQRT1_2	0.70710678118654752440	1/sqrt(2)
M_LNPI	1.14472988584940017414	log_e(pi) [4.0.2]
M_EULER	0.57721566490153286061	Constante de Euler [4.0.2]

Solamente M_PI está disponible en versiones de PHP anteriores a PHP 4.0.0, ésta inclusive. Todas las otras constantes están disponibles a partir de PHP 4.0.0. Las constantes con la etiqueta [4.0.2] fueron añadidas en PHP 4.0.2.

Tabla de contenidos

exp -- e elevado a...
expm1 -- Devuelve exp(number) - 1, computado de una forma que es precisa incluso cuando el valor del número es cercano a cero
floor -- redondea fracciones hacia abajo
fmod -- Devuelve el residuo de punto flotante (módulo) de la división de los argumentos
getrandmax -- Muestra el mayor valor aleatorio posible
HexDec -- hexadecimal a decimal
hypot -- Devuelve sqrt(num1*num1 + num2*num2)
is_finite -- Encuentra si un valor es un número finito legal
is_infinite -- Encuentra si un valor es infinito
is_nan -- Encuentra si un valor no es un número
lcg_value -- Generador lineal congruente combinado
log10 -- Logaritmo en base-10
log1p -- Devuelve log(1 + numero), computado en una forma que es precisa incluso cuando el valor del número es cercano a cero
log -- Logaritmo natural
max -- encuentra el valor mayor
min -- encuentra el valor menor
mt_getrandmax -- muestra el mayor valor aleatorio posible
mt_rand -- genera un valor aleatorio mejorado
mt_srand -- Introduce la semilla del generador de números aleatorios mejorado
OctDec -- octal a decimal
pi -- devuelve el valor de pi
pow -- expresión exponencial
rad2deg -- Convierte el número en radianes a su ñalente en grados
rand -- genera un valor aleatorio
round -- Redondea un float.
sin -- seno
sinh -- Seno hiperbólico
sqrt -- Raíz cuadrada
srand -- introduce la semilla del generador de números aleatorios
tan -- tangente
tanh -- Tangente hiperbólica

abs

(PHP 3, PHP 4 , PHP 5)

abs -- Valor absoluto

Descripción

mixed **abs** (mixed number)

Devuelve el valor absoluto de un número. Si el número del argumento es decimal, el tipo de retorno también es decimal, de otra manera devuelve un entero.

acos

(PHP 3, PHP 4 , PHP 5)

acos -- Arco coseno

Descripción

float **acos** (float arg)

Devuelve el arco coseno de arg en radianes.

Vea también **asin()** y **atan()**.

acosh

(PHP 4 >= 4.1.0, PHP 5)

acosh -- Coseno hiperbólico inverso

Descripción

float **acosh** (float arg)

Devuelve el coseno hiperbólico inverso de *arg*, esto es, el valor de cuyo coseno hiperbólico es *arg*.

> **Nota:** Esta función no está implementada en plataformas Windows.

Vea también: **acos()**, **asinh()**, y **atanh()**.

asin

(PHP 3, PHP 4 , PHP 5)

asin -- Arco seno

Descripción

float **asin** (float arg)

Devuelve el arco seno de arg en radianes.

Vea también **acos()** y **atan()**.

asinh

(PHP 4 >= 4.1.0, PHP 5)

asinh -- Seno hiperbólico inverso

Descripción

float **asinh** (float arg)

Devuelve el seno hiperbólico inverso de *arg*, esto es, el valor de cuyo seno hiperbólico es *arg*.

Nota: Esta función no está implementada en plataformas Windows.

Vea también: **asin()**, **acosh()**, y **atanh()**.

atan2

(PHP 3>= 3.0.5, PHP 4 , PHP 5)

atan2 -- Arco tangente de dos variables

Descripción

float **atan2** (float y, float x)

Esta función calcula la arco tangente de las dos variables x e y. Es similar a el cálculo de la arco tangente de y / x, excepto que los signos de ambos argumentos son usados para determinar el cuadrante del resultado

La función devuelve el resultado en radianes, el cual está entre -PI y PI (inclusive).

Vea también **acos()** y **atan()**.

atan

(PHP 3, PHP 4 , PHP 5)

atan -- Arco tangente

Descripción

float **atan** (float arg)

Devuelve la arco tangente de arg en radianes.

Vea también **acos()** y **atan()**.

atanh

(PHP 4 >= 4.1.0, PHP 5)

atanh -- Tangente hiperbólica inversa

Descripción

float **atanh** (float arg)

Devuelve la tangente hiperbólica inversa de *arg*, esto es, el valor de cuya tangente hiperbólica es *arg*.

Nota: Esta función no está implementada en plataformas Windows.

Vea también: **atan()**, **asinh()**, y **acosh()**.

base_convert

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

base_convert -- Convierte un número entre bases arbitrarias

Descripción

strin **base_convert** (string number, int frombase, int tobase)

Devuelve una cadena conteniendo el *number* representado en base *tobase*. La base en la cual *number* es dado viene especificada en *frombase*. Tanto *frombase* y *tobase* tienen que estar entre 2 y 36, inclusive. Los dígitos en números con una base mayor que 10 serán representados con las letras a-z, a vale 10, b vale 11 y z vale 36.

Ejemplo 1. base_convert()

```
$binary = base_convert($hexadecimal, 16, 2);
```

BinDec

(PHP 3, PHP 4 , PHP 5)

BinDec -- binario a decimal

Descripción

int **bindec** (string binary_string)

Devuelve el ñalente decimal del número binario representado por el argumento binary_string.

BinDec convierte un número binario en un número decimal. El mayor número que puede ser convertido son 31 bits de unos o 2147483647 en decimal.

Vea también la función **decbin()** .

ceil

(PHP 3, PHP 4 , PHP 5)

ceil -- Redondea fracciones hacia arriba

Descripción

int **ceil** (float number)

Devuelve el valor entero superior más cercano a *number*. El uso de **ceil()** con enteros es una perdida

de tiempo absoluta.

NOTA: PHP/FI 2's **ceil()** devuelve un float. Use: *$new = (double)ceil($number);* para tener el comportamiento antiguo.

Vea también **floor()** y **round()**.

cos

(PHP 3, PHP 4 , PHP 5)

cos -- coseno

Descripción

float **cos** (float arg)

Devuelve el coseno de arg en radianes.

Vea también **sin()** y **tan()**.

cosh

(PHP 4 >= 4.1.0, PHP 5)

cosh -- Coseno hiperbólico

Descripción

float **cosh** (float arg)

Devuelve el coseno hiperbólico de *arg*, definido como *(exp(arg) + exp(-arg))/2*.

Vea también: **cos()**, **acosh()**, **sin()**, y **tan()**.

DecBin

(PHP 3, PHP 4 , PHP 5)

DecBin -- decimal a binario

Descripción

string **decbin** (int number)

Devuelve una cadena conteniendo la representación en binario de el número dado en el argumento. El número mayor que puede ser convertido es 2147483647 en decimal que resulta una cadena de 31 unos.

Vea también la función **bindec()** .

DecHex

(PHP 3, PHP 4 , PHP 5)

DecHex -- decimal a hexadecimal

Descripción

string **dechex** (int number)

Devuelve una cadena conteniendo la representación hexadecimal del número dado en el argumento. El mayor número que puede ser convertido es 2147483647 en decimal resultando "7fffffff".

Vea también la función **hexdec()** .

DecOct

(PHP 3, PHP 4 , PHP 5)

DecOct -- decimal a octal

Descripción

string **decoct** (int number)

Devuelve una cadena conteniendo la representación octal del número dado en el argumento. Returns a string containing an octal representation of the given number argument. El mayor número que puede ser ocnvertido es 2147483647 en decimal resultando "17777777777". Vea también **octdec()**.

deg2rad

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

deg2rad -- Convierte el número en grados a su ñalente en radianes

Descripción

float **deg2rad** (float numero)

Esta función convierte a *numero* desde grados a su ñalente en radianes.

Ejemplo 1. Ejemplo de deg2rad()

```
<?php

echo deg2rad(45); // 0.785398163397
var_dump(deg2rad(45) === M_PI_4); // bool(true)

?>
```

Vea también **rad2deg()**.

exp

(PHP 3, PHP 4 , PHP 5)

exp -- e elevado a...

Descripción

float **exp** (float arg)

Devuelve el número e elevado a la potencia de *arg*.

Vea también **pow()**.

expm1

(PHP 4 >= 4.1.0, PHP 5)

expm1 -- Devuelve exp(number) - 1, computado de una forma que es precisa incluso cuando el valor del número es cercano a cero

Descripción

float **expm1** (float numero)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

Nota: Esta función no está implementada en plataformas Windows.

floor

(PHP 3, PHP 4 , PHP 5)

floor -- redondea fracciones hacia abajo

Descripción

float **floor** (float number)

Devuelve el valor entero inferior más cercano a *number*. El uso de **floor()** con enteros es una

perdida de tiempo absoluta.

NOTA: PHP/FI 2's **floor()** devuelve un float. Use: *$new = (double)floor($number);* para tener el comportamiento antiguo.

Vea también **ceil()** y **round()**.

fmod

(PHP 4 >= 4.2.0, PHP 5)

fmod -- Devuelve el residuo de punto flotante (módulo) de la división de los argumentos

Descripción

float **fmod** (float x, float y)

Devuelve el residuo de punto flotante de dividir el dividendo (*x*) por el divisor (*y*). El residuo (*r*) es definido como: x = i * y + r, para algún entero *i*. Si *y* es diferente de cero, *r* tiene el mismo signo que *x* y una magnitud menor que la magnitud de *y*.

Ejemplo 1. Uso de fmod()

```
<?php
$x = 5.7;
$y = 1.3;
$r = fmod($x, $y);
// $r es igual a 0.5, ya que 4 * 1.3 + 0.5 = 5.7
?>
```

getrandmax

(PHP 3, PHP 4 , PHP 5)

getrandmax -- Muestra el mayor valor aleatorio posible

Descripción

int **getrandmax** (void)

Devuelve el valor máximo que puede devolver una llamada a **rand()**.

Vea también **rand()**, **srand()** **mt_rand()**, **mt_srand()** y **mt_getrandmax()**.

HexDec

(PHP 3, PHP 4 , PHP 5)

HexDec -- hexadecimal a decimal

Descripción

int **hexdec** (string hex_string)

Devuelve el ñalente decimal de el número hexadecimal representado por el argumento hex_string. HexDec convierte una cadena hexadecimal en un número decimal. El número mayor que puede ser convertido es 7fffffff o 2147483647 en decimal.

Vea también la función **dechex()** .

hypot

(PHP 4 >= 4.1.0, PHP 5)

hypot -- Devuelve sqrt(num1*num1 + num2*num2)

Descripción

float **hypot** (float num1, float num2)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

is_finite

(PHP 4 >= 4.2.0, PHP 5)

is_finite -- Encuentra si un valor es un número finito legal

Descripción

bool **is_finite** (float val)

Devuelve **TRUE** si *val* es un número finito legal entre el rango permitido para un flotante PHP en esta plataforma.

Vea también **is_infinite()** y **is_nan()**.

is_infinite

(PHP 4 >= 4.2.0, PHP 5)

is_infinite -- Encuentra si un valor es infinito

Descripción

bool **is_infinite** (float val)

Devuelve **TRUE** si *val* es infinito (positivo o negativo), como el resultado de *log(0)* o cualquier valor demasiado grande para caber en un flotante en esta plataforma.

Vea también **is_finite()** y **is_nan()**.

is_nan

(PHP 4 >= 4.2.0, PHP 5)

is_nan -- Encuentra si un valor no es un número

Descripción

bool **is_nan** (float val)

Devuelve **TRUE** si *val* no es un número ('not a number'), como el resultado de *acos(1.01)*.

Vea también **is_finite()** y **is_infinite()**.

lcg_value

(PHP 4 , PHP 5)

lcg_value -- Generador lineal congruente combinado

Descripción

float **lcg_value** (void)

lcg_value() devuelve un número pseudo-aleatorio en el rango (0, 1). La función combina dos generadores congruentes con periodos de 2^31 - 85 y 2^31 - 249. El periodo de esta función es igual al producto de ambos primos.

log10

(PHP 3, PHP 4 , PHP 5)

log10 -- Logaritmo en base-10

Descripción

float **log10** (float arg)

Devuelve el logaritmo en base-10 de arg.

log1p

(PHP 4 >= 4.1.0, PHP 5)

log1p -- Devuelve log(1 + numero), computado en una forma que es precisa incluso cuando el valor del número es cercano a cero

Descripción

float **log1p** (float numero)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

Nota: Esta función no está implementada en plataformas Windows.

log

(PHP 3, PHP 4 , PHP 5)

log -- Logaritmo natural

Descripción

float **log** (float arg)

Devuelve el logaritmo de arg.

max

(PHP 3, PHP 4 , PHP 5)

max -- encuentra el valor mayor

Descripción

mixed **max** (mixed arg1, mixed arg2, mixed argn)

max() devuelve el número mayor de los valores de los parámetros.

Si el primer parámetro es un array, **max()** devuelve el mayor valor en ese array. Si el primer parámetro es un entero, string o double, necesita al menos dos parámetros y **max()** devuelve el mayor número de estos valores. Puede comparar un número ilimitado de valores.

Si uno o más de los valores es un double, todos los valores serán tratados como doubles, y devolverá un double. Si ninguno de los valores es un double, todos ellos serán tratados como enteros, y se devolverá un entero.

min

(PHP 3, PHP 4 , PHP 5)

min -- encuentra el valor menor

Descripción

mixed **min** (mixed arg1, mixed arg2, mixed argn)

min() returns the numerically lowest of the parameter values.

Si el primer parámetro es un array, **min()** devuelve el menor valor en ese array. Si el primer parámetro es un entero, string o double, necesita al menos dos parámetros y **min()** devuelve el número menor de estos valores. Puede comparar un número ilimitado de valores.

Si uno o más de los valores es un double, todos los valores serán tratados como doubles, y devolverá un double. Si ninguno de los valores es un double, todos ellos serán tratados como enteros, y se devolverá un entero.

mt_getrandmax

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

mt_getrandmax -- muestra el mayor valor aleatorio posible

Descripción

int **mt_getrandmax** (void)

Devuelve el valor máximo que se puede obtener con una llamada a **mt_rand()**.

Vea también **mt_rand()**, **mt_srand()** **rand()**, **srand()** y **getrandmax()**.

mt_rand

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

mt_rand -- genera un valor aleatorio mejorado

Descripción

int **mt_rand** ([int min [, int max]])

Muchos generadores de números aleatorios de libcs antiguas tienen características dudosas o desconocidas y son lentas. Por defecto, PHP usa el generador de números aleatorios de libc con la función **rand()**. La función **mt_rand()** es un reemplazo para esta. Usa un generador de números aleatorios con características conocidas, el Tornado de Mersenne, que es capaz de producir números aleatorios que incluso se pueden emplear para propósitos criptográficos y es cuatro veces más rápido que la media de los que provee libc. La página principal del Tornado de Mersenne puede

encontrarse en http://www.math.keio.ac.jp/~matumoto/emt.html, y una versión optimizada del código del TM esta disponible en http://www.scp.syr.edu/~marc/hawk/twister.html.

Si es llamada sin los parámetros opcionales min y max **mt_rand()** devuelve un valor pseudo-aleatorio entre 0 y RAND_MAX. Si quiere un número aleatorio entre 5 y 15 (inclusive), use mt_rand(5,15).

Recuerde introducir la semilla del generador de números aleatorios antes de usar la instrucción con **mt_srand()**.

Vea también **mt_srand()**, **mt_getrandmax()**, **srand()**, **rand()** y **getrandmax()**.

mt_srand

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

mt_srand -- Introduce la semilla del generador de números aleatorios mejorado

Descripción

void **mt_srand** (int seed)

Introduce la semilla *seed* en el generador de números aleatorios mejorado.

```
// seed son los microsegundos desde el último segundo "entero"
mt_srand((double)microtime()*1000000);
$randval = mt_rand();
```

Vea también **mt_rand()**, **mt_getrandmax()**, **srand()**, **rand()** y **getrandmax()**.

OctDec

(PHP 3, PHP 4 , PHP 5)

OctDec -- octal a decimal

Descripción

int **octdec** (string octal_string)

Devuelve el ñalente decimal del número octal representado por el argumento octal_string. OctDec convierte una cadena octal en un número decimal. El mayor número que puede ser convertido es 17777777777 o 2147483647 en decimal.

Vea también **decoct()**.

pi

(PHP 3, PHP 4 , PHP 5)

pi -- devuelve el valor de pi

Descripción

double **pi** (void)

Devuelve una aproximación de pi.

pow

(PHP 3, PHP 4 , PHP 5)

pow -- expresión exponencial

Descripción

float **pow** (float base, float exp)

Devuelve base elevado a la potencia de exp.

Vea también **exp()**.

rad2deg

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

rad2deg -- Convierte el número en radianes a su ñalente en grados

Descripción

float **rad2deg** (float numero)

Esta función convierte a *numero* desde radianes a grados.

Ejemplo 1. Ejemplo de rad2deg()

```
<?php

echo rad2deg(M_PI_4); // 45

?>
```

Vea también **deg2rad()**.

rand

(PHP 3, PHP 4 , PHP 5)

rand -- genera un valor aleatorio

Descripción

int **rand** ([int min [, int max]])

Si es llamada sin los argumentos opcionales min y max, rand() devuelve un valor pseudo-aleatorio entre 0 y RAND_MAX. Si quiere un número aleatorio entre 5 y 15 (inclusive), por ejemplo, use rand(5,15).

Recuerde introducir la semilla del generador de números aleatorios antes de usar **srand()**.

Vea también **srand()**, **getrandmax()**, **mt_rand()**, **mt_srand()** y **mt_getrandmax()**.

round

(PHP 3, PHP 4 , PHP 5)

round -- Redondea un float.

Descripción

double **round** (double val)

Devuelve el valor redondeado de *val*.

```
$foo = round( 3.4 );   // $foo == 3.0
$foo = round( 3.5 );   // $foo == 4.0
$foo = round( 3.6 );   // $foo == 4.0
```

Vea también **ceil()** y **floor()**.

sin

(PHP 3, PHP 4 , PHP 5)

sin -- seno

Descripción

float **sin** (float arg)

Devuelve el seno de arg en radianes.

Vea también **cos()** y **tan()**.

sinh

(PHP 4 >= 4.1.0, PHP 5)

sinh -- Seno hiperbólico

Descripción

float **sinh** (float arg)

Devuelve el seno hiperbólico de *arg*, definido como *(exp(arg) - exp(-arg))/2*.

Vea también: **sin()**, **asinh()**, **cos()**, y **tan()**.

sqrt

(PHP 3, PHP 4 , PHP 5)

sqrt -- Raíz cuadrada

Descripción

float **sqrt** (float arg)

Devuelve la raíz cuadrada de arg.

srand

(PHP 3, PHP 4 , PHP 5)

srand -- introduce la semilla del generador de números aleatorios

Descripción

void **srand** (int seed)

Inicializa el generador de números aleatorios con *seed*.

```
// seed son los microsegundos desde el último segundo "entero"
srand((double)microtime()*1000000);
$randval = rand();
```

Vea también **rand()**, **getrandmax()**, **mt_rand()**, **mt_srand()** y **mt_getrandmax()**.

tan

(PHP 3, PHP 4 , PHP 5)

tan -- tangente

Descripción

float **tan** (float arg)

Devuelve la tangente de arg en radianes.

Vea también **sin()** y **cos()**.

tanh

(PHP 4 >= 4.1.0, PHP 5)

tanh -- Tangente hiperbólica

Descripción

float **tanh** (float arg)

Devuelve la tangente hiporbólica de *arg*, definida como *sinh(arg)/cosh(arg)*.

Vea también: **tan()**, **atanh()**, **sin()**, y **cos()**.

LXIV. MaxDB PHP Extension

Introducción

The MaxDB PHP extension allows you to access the functionality provided by MaxDB 7.5.0 and above. More information about the MaxDB Database server can be found at http://www.mysql.com/products/maxdb/.

Documentation for MaxDB can be found at http://dev.mysql.com/doc/maxdb/.

Requirimientos

In order to have these functions available, you must compile PHP with MaxDB support.Additionally, you must have the MaxDB SQLDBC runtime library available to access the MaxDB server.

Documentation for MaxDB SQLDBC can be found at http://dev.mysql.com/doc/maxdb/.

Download the MaxDB SQLDBC package from http://dev.mysql.com/downloads/maxdb/clients.html.

Instalación

By using the *--with-maxdb[=DIR]* configuration option you enable PHP to access MaxDB databases. *[DIR]* points to the directory that contains the installed MaxDB SQLDBC package.

Windows users will need to enable `php_maxdb.dll` inside of `php.ini`.

Configuración en tiempo de ejecución

El comportamiento de estas funciones está afectado por los valores definidos en `php.ini`.

Tabla 1. MaxDB Configuration Options

Name	Default	Changeable
maxdb.default_host	NULL	PHP_INI_ALL
maxdb.default_db	NULL	PHP_INI_ALL
maxdb.default_user	NULL	PHP_INI_ALL
maxdb.default_password	NULL	PHP_INI_ALL
maxdb.long_readlen	NULL	PHP_INI_ALL

For further details and definitions of the PHP_INI_* constants, see the Apéndice H.

A continuación se presenta una corta explicación de las directivas de configuración.

maxdb.default_host **string**

The default server host to use when connecting to the database server if no other host is specified.

maxdb.default_db **string**

The default server database to use when connecting if no other database is specified.

maxdb.default_user **string**

The default user name to use when connecting to the database server if no other name is specified.

maxdb.default_password **string**

The default password to use when connecting to the database server if no other password is specified.

maxdb.long_readlen **integer**

The default maximum length of bytes that is transferred to the client if long data is retrieved from the MaxDB database server.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

The following constants to use with **maxdb_options()** are defined. For further description of these constants see http://dev.mysql.com/doc/maxdb/.

Tabla 2. MaxDB PHP client constants

Constant	Description
MAXDB_COMPNAME	The component name used to initialise the SQLDBC runtime environment.
MAXDB_APPLICATION	The application to be connected to the database.
MAXDB_APPVERSION	The version of the application.
MAXDB_SQLMODE	The SQL mode.
MAXDB_UNICODE	TRUE, if the connection is an unicode (UCS2) client or FALSE, if not.
MAXDB_TIMEOUT	The maximum allowed time of inactivity after which the connection to the database is closed by the system.
MAXDB_ISOLATIONLEVEL	Specifies whether and how shared locks and exclusive locks are implicitly requested or released.
MAXDB_PACKETCOUNT	The number of different request packets used for the connection.
MAXDB_STATEMENTCACHESIZE	The number of prepared statements to be cached for the connection for re-use.
MAXDB_CURSORPREFIX	The prefix to use for result tables that are automatically named.

The function **maxdb_fetch_array()** uses a constant for the different types of result arrays. The following constants are defined:

Tabla 3. MaxDB fetch constants

Constant	Description
MAXDB_ASSOC	Columns are returned into the array having the fieldname as the array index.
MAXDB_BOTH	Columns are returned into the array having both a numerical index and the fieldname as the array index.
MAXDB_NUM	Columns are returned into the array having a numerical index to the fields. This index starts with 0, the first field in the result.

Ejemplos

All examples in the MaxDB PHP documentation use the HOTELDB demo database from MaxDB. More about this database can be found at http://dev.mysql.com/doc/maxdb/en/98/11b83fa6b33c17e10000000a114084/frameset.htm.

This simple example shows how to connect, execute a query, print resulting rows and disconnect from a MaxDB database.

Ejemplo 1. MaxDB extension overview example

```
<?php
/* Connecting, selecting database */
$link = maxdb_connect("maxdb_host", "maxdb_user", "maxdb_password")
    or die("Could not connect : " . maxdb_connect_error());
echo "Connected successfully";
maxdb_select_db("my_database") or die("Could not select database");

/* Performing SQL query */
$query = "SELECT * FROM my_table";
$result = maxdb_query($link, $query) or die("Query failed : " . maxdb_error());

/* Printing results in HTML */
echo "<table>\n";
while ($line = maxdb_fetch_array($result, MAXDB_ASSOC)) {
    echo "\t<tr>\n";
    foreach ($line as $col_value) {
        echo "\t\t<td>$col_value</td>\n";
    }
    echo "\t</tr>\n";
}
echo "</table>\n";

/* Free resultset */
maxdb_free_result($result);

/* Closing connection */
maxdb_close($link);
?>
```

Tabla de contenidos

maxdb_fetch_resource -- Returns the current row of a result set as an resource
maxdb_fetch_row -- Get a result row as an enumerated array
maxdb_fetch -- Alias for **maxdb_stmt_fetch()**
maxdb_field_count -- Returns the number of columns for the most recent query
maxdb_field_seek -- Set result pointer to a specified field offset
maxdb_field_tell -- Get current field offset of a result pointer
maxdb_free_result -- Frees the memory associated with a result
maxdb_get_client_info -- Returns the MaxDB client version as a string
maxdb_get_client_version -- Get MaxDB client info
maxdb_get_host_info -- Returns a string representing the type of connection used
maxdb_get_metadata -- Alias for **maxdb_stmt_result_metadata()**
maxdb_get_proto_info -- Returns the version of the MaxDB protocol used
maxdb_get_server_info -- Returns the version of the MaxDB server
maxdb_get_server_version -- Returns the version of the MaxDB server as an integer
maxdb_info -- Retrieves information about the most recently executed query
maxdb_init -- Initializes MaxDB and returns an resource for use with maxdb_real_connect
maxdb_insert_id -- Returns the auto generated id used in the last query
maxdb_kill -- Disconnects from a MaxDB server
maxdb_master_query -- Enforce execution of a query on the master in a master/slave setup
maxdb_more_results -- Check if there any more query results from a multi query
maxdb_multi_query -- Performs a query on the database
maxdb_next_result -- Prepare next result from multi_query
maxdb_num_fields -- Get the number of fields in a result
maxdb_num_rows -- Gets the number of rows in a result
maxdb_options -- Set options
maxdb_param_count -- Alias for **maxdb_stmt_param_count()**
maxdb_ping -- Pings a server connection, or tries to reconnect if the connection has gone down
maxdb_prepare -- Prepare a SQL statement for execution
maxdb_query -- Performs a query on the database
maxdb_real_connect -- Opens a connection to a MaxDB server
maxdb_real_escape_string -- Escapes special characters in a string for use in a SQL statement, taking into account the current charset of the connection
maxdb_real_query -- Execute an SQL query
maxdb_report -- Enables or disables internal report functions
maxdb_rollback -- Rolls back current transaction
maxdb_rpl_parse_enabled -- Check if RPL parse is enabled
maxdb_rpl_probe -- RPL probe
maxdb_rpl_query_type -- Returns RPL query type
maxdb_select_db -- Selects the default database for database queries
maxdb_send_long_data -- Alias for **maxdb_stmt_send_long_data()**
maxdb_send_query -- Send the query and return
maxdb_server_end -- Shut down the embedded server
maxdb_server_init -- Initialize embedded server
maxdb_set_opt -- Alias of **maxdb_options()**
maxdb_sqlstate -- Returns the SQLSTATE error from previous MaxDB operation
maxdb_ssl_set -- Used for establishing secure connections using SSL
maxdb_stat -- Gets the current system status
maxdb_stmt_affected_rows -- Returns the total number of rows changed, deleted, or inserted by the last executed statement
maxdb_stmt_bind_param -- Binds variables to a prepared statement as parameters
maxdb_stmt_bind_result -- Binds variables to a prepared statement for result storage
maxdb_stmt_close_long_data -- Ends a sequence of **maxdb_stmt_send_long_data()**
maxdb_stmt_close -- Closes a prepared statement
maxdb_stmt_data_seek -- Seeks to an arbitray row in statement result set
maxdb_stmt_errno -- Returns the error code for the most recent statement call

maxdb_affected_rows

(no version information, might be only in CVS)

maxdb_affected_rows -- Gets the number of affected rows in a previous MaxDB operation

Description

Procedural style:

mixed **maxdb_affected_rows** (resource link)

maxdb_affected_rows() returns the number of rows affected by the last INSERT, UPDATE, or DELETE query associated with the provided *link* parameter. If this number cannot be determined, this function will return -1.

> **Nota:** For SELECT statements **maxdb_affected_rows()** works like **maxdb_num_rows ()**.

The **maxdb_affected_rows()** function only works with queries which modify a table. In order to return the number of rows from a SELECT query, use the **maxdb_num_rows()** function instead.

Return Values

An integer greater than zero indicates the number of rows affected or retrieved. Zero indicates that no records where updated for an UPDATE statement, no rows matched the WHERE clause in the query or that no query has yet been executed. -1 indicates that the number of rows affected can not be determined.

See also

maxdb_num_rows(), **maxdb_info()**.

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

if (!$link) {
   printf("Can't connect to localhost. Error: %s\n", maxdb_connect_error());
   exit();
}

maxdb_report (MAXDB_REPORT_OFF);
maxdb_query($link,"DROP TABLE mycustomer");
maxdb_report (MAXDB_REPORT_ERROR);

/* Insert rows */
maxdb_query($link, "CREATE TABLE mycustomer AS SELECT * from hotel.customer");
printf("Affected rows (INSERT): %d\n", maxdb_affected_rows($link));

maxdb_query($link, "ALTER TABLE mycustomer ADD Status int default 0");

/* update rows */
maxdb_query($link, "UPDATE mycustomer SET Status=1 WHERE cno > 50");
printf("Affected rows (UPDATE): %d\n", maxdb_affected_rows($link));

/* delete rows */
maxdb_query($link, "DELETE FROM mycustomer WHERE cno < 50");
printf("Affected rows (DELETE): %d\n", maxdb_affected_rows($link));

/* select all rows */
$result = maxdb_query($link, "SELECT title FROM mycustomer");
printf("Affected rows (SELECT): %d\n", maxdb_affected_rows($link));

maxdb_free_result($result);

/* Delete table Language */
maxdb_query($link, "DROP TABLE mycustomer");

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
Affected rows (INSERT): 15
Affected rows (UPDATE): 15
Affected rows (DELETE): 0
Affected rows (SELECT): -1
```

maxdb_autocommit

(no version information, might be only in CVS)

maxdb_autocommit -- Turns on or off auto-commiting database modifications

Description

Procedural style:

bool **maxdb_autocommit** (resource link, bool mode)

maxdb_autocommit() is used to turn on or off auto-commit mode on queries for the database

connection represented by the *link* resource.

Return values

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

See also

maxdb_commit(), **maxdb_rollback()**.

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

if (!$link) {
   printf("Can't connect to localhost. Error: %s\n", maxdb_connect_error());
   exit();
}

/* turn autocommit on */
maxdb_autocommit($link, TRUE);

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce no output.

maxdb_bind_param

maxdb_bind_param -- Alias for **maxdb_stmt_bind_param()**

Description

This function is an alias of **maxdb_stmt_bind_param()**. For a detailled descripton see description of **maxdb_stmt_bind_param()**.

> **Nota: maxdb_bind_param()** is deprecated and will be removed.

See also

maxdb_stmt_bind_param()

maxdb_bind_result

maxdb_bind_result -- Alias for **maxdb_stmt_bind_result()**

Description

This function is an alias of **maxdb_stmt_bind_result()**. For a detailled descripton see description of **maxdb_stmt_bind_result()**.

> **Nota: maxdb_bind_result()** is deprecated and will be removed.

See also

maxdb_stmt_bind_result()

maxdb_change_user

(no version information, might be only in CVS)

maxdb_change_user -- Changes the user of the specified database connection

Description

Procedural style:

bool **maxdb_change_user** (resource link, string user, string password, string database)

maxdb_change_user() is used to change the user of the specified database connection as given by the *link* parameter and to set the current database to that specified by the *database* parameter.

In order to successfully change users a valid *username* and *password* parameters must be provided and that user must have sufficient permissions to access the desired database. If for any reason authorization fails, the current user authentication will remain.

> **Nota:** Using this command will always cause the current database connection to behave as if was a completely new database connection, regardless of if the operation was completed successfully. This reset includes performing a rollback on any active transactions, closing all temporary tables, and unlocking all locked tables.

Return Values

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

See also:

maxdb_connect() **maxdb_select_db()**

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (!$link) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

if ($result = maxdb_query($link, "SELECT * FROM dual")) {
   $row = maxdb_fetch_row($result);
   printf("Result: %s\n", $row[0]);
   maxdb_free_result($result);
}

/* reset all and select a new database */
if (!maxdb_change_user($link, $user, $passwd, "XXXXXXX")) {
  printf("Database not running\n");
} else {
  printf("Database running\n");
}

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
Result: a
PHP Warning:  maxdb_change_user(): -10709 Connection failed <...>
Database not running
PHP Warning:  maxdb_close(): -10821 Session not connected <...>
```

maxdb_character_set_name

(no version information, might be only in CVS)

maxdb_character_set_name -- Returns the default character set for the database connection

Description

Procedural style:

string **maxdb_character_set_name** (resource link)

Returns the current character set for the database connection specified by the *link* parameter.

Return values

The default character set for the current connection, either ascii or unicode.

See also

maxdb_client_encoding(). **maxdb_real_escape_string()**.

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (!$link) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

/* Print current character set */
$charset = maxdb_character_set_name($link);
printf ("Current character set is %s\n",$charset);

/* close connection */
maxdb_close($link);
?>
```

The above examples would be produce the following output:

```
Current character set is ascii
```

maxdb_client_encoding

maxdb_client_encoding -- Alias of **maxdb_character_set_name()**

Description

This function is an alias of **maxdb_character_set_name()**. For a detailled descripton see description of **maxdb_character_set_name()**.

See also

maxdb_client_encoding(). **maxdb_real_escape_string()**.

maxdb_close_long_data

maxdb_close_long_data -- Alias for **maxdb_stmt_close_long_data()**

Description

This function is an alias of **maxdb_stmt_close_long_data()**. For a detailled descripton see description of **maxdb_stmt_close_long_data()**.

> **Nota: maxdb_close_long_data()** is deprecated and will be removed.

See also

maxdb_stmt_close_long_data()

maxdb_close

(no version information, might be only in CVS)

maxdb_close -- Closes a previously opened database connection

Description

Procedural style:

bool **maxdb_close** (resource link)

The **maxdb_close()** function closes a previously opened database connection specified by the *link* parameter.

Return values

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

See also

maxdb_connect(), **maxdb_init()**, **maxdb_real_connect()**.

maxdb_commit

(no version information, might be only in CVS)

maxdb_commit -- Commits the current transaction

Description

Procedural style:

bool **maxdb_commit** (resource link)

Commits the current transaction for the database connection specified by the *link* parameter.

Return values

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

See also

maxdb_autocommit(), **maxdb_rollback()**.

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (!$link) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

/* set autocommit to off */
maxdb_autocommit($link, FALSE);

maxdb_report (MAXDB_REPORT_OFF);
maxdb_query($link,"DROP TABLE mycustomer");
maxdb_report (MAXDB_REPORT_ERROR);

maxdb_query($link, "CREATE TABLE mycustomer LIKE hotel.customer");

/* Insert some values */
maxdb_query($link, "INSERT INTO mycustomer VALUES (3000,'Mrs','Jenny','Porter','10580',
maxdb_query($link, "INSERT INTO mycustomer VALUES (3100,'Mr','Peter','Brown','48226','1

/* commit transaction */
maxdb_commit($link);

/* close connection */
maxdb_close($link);
?>
```

The above examples produces no output.

maxdb_connect_errno

(no version information, might be only in CVS)

maxdb_connect_errno -- Returns the error code from last connect call

Description

int **maxdb_connect_errno** (void)

The **maxdb_connect_errno()** function will return the last error code number for last call to **maxdb_connect()**. If no errors have occured, this function will return zero.

Return values

An error code value for the last call to **maxdb_connect()**, if it failed. zero means no error occurred.

See also

maxdb_connect(), **maxdb_connect_error()**, **maxdb_errno()**, **maxdb_error()**, **maxdb_sqlstate()**

Example

Ejemplo 1. maxdb_connect_errno sample

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

if (!$link) {
   printf("Can't connect to localhost. Errorcode: %d\n", maxdb_connect_errno());
}
?>
```

The above examples would produce the following output:

```
Affected rows (INSERT): 15
Affected rows (UPDATE): 15
Affected rows (DELETE): 0
Affected rows (SELECT): -1
```

maxdb_connect_error

(no version information, might be only in CVS)

maxdb_connect_error -- Returns a string description of the last connect error

Description

string **maxdb_connect_error** (void)

The **maxdb_connect_error()** function is identical to the corresponding **maxdb_connect_errno()** function in every way, except instead of returning an integer error code the **maxdb_connect_error()** function will return a string representation of the last error to occur for the last **maxdb_connect()** call. If no error has occured, this function will return an empty string.

Return values

A string that describes the error. An empty string if no error occurred.

See also

maxdb_connect(), **maxdb_connect_errno()**, **maxdb_errno()**, **maxdb_error()**, **maxdb_sqlstate()**

Example

Ejemplo 1. maxdb_connect_error sample

```
<?php

$link = maxdb_connect("localhost", "nonexisting_user", "");

if (!$link) {
   printf("Can't connect to localhost. Error: %s\n", maxdb_connect_error());
}
?>
```

The above examples would produce the following output:

```
PHP Warning:  maxdb_connect(): -4008 POS(1) Unknown user name/password combination <...
Can't connect to localhost. Error: POS(1) Unknown user name/password combination
```

maxdb_connect

(no version information, might be only in CVS)

maxdb_connect -- Open a new connection to the MaxDB server

Description

Procedural style

resource **maxdb_connect** ([string host [, string username [, string passwd [, string dbname [, int port [, string socket]]]]]])

The **maxdb_connect()** function attempts to open a connection to the MaxDB Server running on *host* which can be either a host name or an IP address. Passing the string "localhost" to this parameter, the local host is assumed. If successful, the **maxdb_connect()** will return an resource representing the connection to the database, or **FALSE** on failure.

The *username* and *password* parameters specify the username and password under which to connect to the MaxDB server. If the password is not provided (the **NULL** value is passed), the MaxDB server will attempt to authenticate the user against the *maxdb.default_pw* in `php.ini`.

The *dbname* parameter if provided will specify the default database to be used when performing queries. If not provied, the entry *maxdb.default_db* in `php.ini` is used.

The *port* and *socket* parameters are ignored for the MaxDB server.

Return values

Returns a resource which represents the connection to a MaxDB Server or **FALSE** if the connection failed.

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (!$link) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

printf("Host information: %s\n", maxdb_get_host_info($link));

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
Host information: localhost
```

maxdb_data_seek

(no version information, might be only in CVS)

maxdb_data_seek -- Adjusts the result pointer to an arbitary row in the result

Description

Procedural style:

bool **maxdb_data_seek** (resource result, int offset)

The **maxdb_data_seek()** function seeks to an arbitrary result pointer specified by the *offset* in the result set represented by *result*. The *offset* parameter must be between zero and the total number of rows minus one (0..**maxdb_num_rows()** - 1).

Return values

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

See also

maxdb_store_result(), **maxdb_fetch_row()**, **maxdb_num_rows()**.

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (!$link) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

$query = "SELECT name, state FROM hotel.city ORDER BY name";

if ($result = maxdb_query($link, $query)) {

   /* seek to row no. 400 */
   maxdb_data_seek($result, 10);

   /* fetch row */
   $row = maxdb_fetch_row($result);

   printf ("City: %s  State: %s\n", $row[0], $row[1]);

   /* free result set*/
   maxdb_free_result($result);
}

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
City: Irvine  State: CA
```

maxdb_dump_debug_info

(no version information, might be only in CVS)

maxdb_dump_debug_info -- Dump debugging information into the log

Description

bool **maxdb_dump_debug_info** (resource link)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

maxdb_debug

(no version information, might be only in CVS)

maxdb_debug -- Performs debugging operations

Description

void **maxdb_debug** (string debug)

Return values

maxdb_debug() doesn't return any value.

Example

Ejemplo 1. Procedural style

```
<?php

/* Create a trace file in '/tmp/client.trace' on the local (client) machine: */
/* NOTE: Does nothing in case of MaxDB */
maxdb_debug("d:t:0,/tmp/client.trace");

?>
```

The above examples produces no output.

maxdb_disable_reads_from_master

(no version information, might be only in CVS)

maxdb_disable_reads_from_master -- Disable reads from master

Description

void **maxdb_disable_reads_from_master** (resource link)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

maxdb_disable_rpl_parse

(no version information, might be only in CVS)

maxdb_disable_rpl_parse -- Disable RPL parse

Description

void **maxdb_disable_rpl_parse** (resource link)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

maxdb_embedded_connect

(no version information, might be only in CVS)

maxdb_embedded_connect -- Open a connection to an embedded MaxDB server

Description

resource **maxdb_embedded_connect** ([string dbname])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

maxdb_enable_reads_from_master

(no version information, might be only in CVS)

maxdb_enable_reads_from_master -- Enable reads from master

Description

void **maxdb_enable_reads_from_master** (resource link)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

maxdb_enable_rpl_parse

(no version information, might be only in CVS)

maxdb_enable_rpl_parse -- Enable RPL parse

Description

void **maxdb_enable_rpl_parse** (resource link)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

maxdb_errno

(no version information, might be only in CVS)

maxdb_errno -- Returns the error code for the most recent function call

Description

Procedural style:

int **maxdb_errno** (resource link)

The **maxdb_errno()** function will return the last error code for the most recent MaxDB function call that can succeed or fail with respect to the database link defined by the *link* parameter. If no errors have occured, this function will return zero.

Return values

An error code value for the last call, if it failed. zero means no error occurred.

See also

maxdb_connect_errno(), maxdb_connect_error(), maxdb_error(), maxdb_sqlstate()

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

if (!maxdb_query($link, "SELECT xxx FROM hotel.city")) {
   printf("Errorcode: %d\n", maxdb_errno($link));
}

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
PHP Warning:  maxdb_query(): -4005 POS(8) Unknown column name:XXX [42000] <...>
Errorcode: -4005
```

maxdb_error

(no version information, might be only in CVS)

maxdb_error -- Returns a string description of the last error

Description

Procedural style:

string **maxdb_error** (resource link)

The **maxdb_error()** function is identical to the corresponding maxdb_errno() function in every way, except instead of returning an integer error code the **maxdb_error()** function will return a string representation of the last error to occur for the database connection represented by the *link* parameter. If no error has occured, this function will return an empty string.

Return values

A string that describes the error. An empty string if no error occurred.

See also

maxdb_connect_errno(), maxdb_connect_error(), maxdb_errno(), maxdb_sqlstate()

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

if (!maxdb_query($link, "SELECT xxx FROM hotel.city")) {
   printf("Errormessgae: %s\n", maxdb_error($link));
}

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
PHP Warning:  maxdb_query(): -4005 POS(8) Unknown column name:XXX [42000]
Errormessgae: POS(8) Unknown column name:XXX
```

maxdb_escape_string

maxdb_escape_string -- Alias of **maxdb_real_escape_string()**

Description

This function is an alias of **maxdb_real_escape_string()**.

maxdb_execute

maxdb_execute -- Alias for **maxdb_stmt_execute()**

Description

This function is an alias of **maxdb_stmt_execute()**. For a detailled descripton see description of **maxdb_stmt_execute()**.

> **Nota: maxdb_execute()** is deprecated and will be removed.

See also

maxdb_stmt_execute()

maxdb_fetch_array

(no version information, might be only in CVS)

maxdb_fetch_array -- Fetch a result row as an associative, a numeric array, or both

Description

Procedural style:

mixed **maxdb_fetch_array** (resource result [, int resulttype])

Returns an array that corresponds to the fetched row or **NULL** if there are no more rows for the resultset represented by the *result* parameter.

maxdb_fetch_array() is an extended version of the **maxdb_fetch_row()** function. In addition to storing the data in the numeric indices of the result array, the **maxdb_fetch_array()** function can also store the data in associative indices, using the field names of the result set as keys.

> **Nota:** Los nombres de campos retornados por esta función diferencian entre *mayusculas y minusculas*.

> **Nota:** Esta funcion define campos NULL como valores PHP **NULL**.

If two or more columns of the result have the same field names, the last column will take precedence and overwrite the earlier data. In order to access multiple columns with the same name, the numerically indexed version of the row must be used.

The optional second argument *resulttype* is a constant indicating what type of array should be produced from the current row data. The possible values for this parameter are the constants MAXDB_ASSOC, MAXDB_NUM, or MAXDB_BOTH. By default the **maxdb_fetch_array()** function will assume MAXDB_BOTH for this parameter.

By using the MAXDB_ASSOC constant this function will behave identically to the **maxdb_fetch_assoc()**, while MAXDB_NUM will behave identically to the **maxdb_fetch_row()** function. The final option MAXDB_BOTH will create a single array with the attributes of both.

Return values

Returns an array that corresponds to the fetched row or **NULL** if there are no more rows in resultset.

See also

maxdb_fetch_assoc(), **maxdb_fetch_row()**, **maxdb_fetch_resource()**.

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

$query = "SELECT name, state FROM hotel.city ORDER by zip";
$result = maxdb_query($link, $query);

/* numeric array */
$row = maxdb_fetch_array($result, MAXDB_NUM);
printf ("%s (%s)\n", $row[0], $row[1]);

/* associative array */
$row = maxdb_fetch_array($result, MAXDB_ASSOC);
printf ("%s (%s)\n", $row["NAME"], $row["STATE"]);

/* associative and numeric array */
$row = maxdb_fetch_array($result, MAXDB_BOTH);
printf ("%s (%s)\n", $row[0], $row["STATE"]);

/* free result set */
maxdb_free_result($result);

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
New York (NY)
New York (NY)
Long Island (NY)
```

maxdb_fetch_assoc

(no version information, might be only in CVS)

maxdb_fetch_assoc -- Fetch a result row as an associative array

Description

Procedural style:

array **maxdb_fetch_assoc** (resource result)

Returns an associative array that corresponds to the fetched row or **NULL** if there are no more rows.

The **maxdb_fetch_assoc()** function is used to return an associative array representing the next row in the result set for the result represented by the *result* parameter, where each key in the array represents the name of one of the result set's columns.

If two or more columns of the result have the same field names, the last column will take precedence. To access the other column(s) of the same name, you either need to access the result with numeric indices by using **maxdb_fetch_row()** or add alias names.

> **Nota:** Los nombres de campos retornados por esta función diferencian entre *mayusculas*

y minusculas.

Nota: Esta funcion define campos NULL como valores PHP **`NULL`**.

Return values

Returns an array that corresponds to the fetched row or **`NULL`** if there are no more rows in resultset.

See also

maxdb_fetch_array(), **maxdb_fetch_row()**, **maxdb_fetch_resource()**.

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

$query = "SELECT name, state FROM hotel.city ORDER by zip";

if ($result = maxdb_query($link, $query)) {

   /* fetch associative array */
   while ($row = maxdb_fetch_assoc($result)) {
       printf ("%s (%s)\n", $row["NAME"], $row["STATE"]);
   }

   /* free result set */
   maxdb_free_result($result);
}

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
New York (NY)
New York (NY)
Long Island (NY)
Albany (NY)
Washington (DC)
Washington (DC)
Washington (DC)
Silver Spring (MD)
Daytona Beach (FL)
Deerfield Beach (FL)
Clearwater (FL)
Cincinnati (OH)
Detroit (MI)
Rosemont (IL)
Chicago (IL)
Chicago (IL)
New Orleans (LA)
Dallas (TX)
Los Angeles (CA)
Hollywood (CA)
Long Beach (CA)
Palm Springs (CA)
Irvine (CA)
Santa Clara (CA)
Portland (OR)
```

maxdb_fetch_field_direct

(no version information, might be only in CVS)

maxdb_fetch_field_direct -- Fetch meta-data for a single field

Description

Procedural style:

mixed **maxdb_fetch_field_direct** (resource result, int fieldnr)

maxdb_fetch_field_direct() returns an resource which contains field definition informations from specified resultset. The value of fieldnr must be in the range from *0* to *number of fields - 1*.

Return values

Returns a resource which contains field definition information or **`FALSE`** if no field information for specified *fieldnr* is available.

Tabla 1. Object attributes

Attribute	Description
name	The name of the column
max_length	The maximum width of the field for the result set.
type	The data type used for this field
decimals	The number of decimals used (for integer fields)

See also

maxdb_num_fields() maxdb_fetch_field() maxdb_fetch_fields()

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

$query = "SELECT name, cno from hotel.customer ORDER BY name";

if ($result = maxdb_query($link, $query)) {

   /* Get field information for column 'cno' */
   $finfo = maxdb_fetch_field_direct($result, 1);

   printf("Name:     %s\n", $finfo->name);
   printf("Table:    %s\n", $finfo->table);
   printf("max. Len: %d\n", $finfo->max_length);
   printf("Flags:    %d\n", $finfo->flags);
   printf("Type:     %d\n", $finfo->type);

   maxdb_free_result($result);
}

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
Name:     CNO
Table:
max. Len: 4
Flags:    -1
Type:     0
```

maxdb_fetch_field

(no version information, might be only in CVS)

maxdb_fetch_field -- Returns the next field in the result set

Description

Procedural style:

mixed **maxdb_fetch_field** (resource result)

The **maxdb_fetch_field()** returns the definition of one column of a result set as an resource. Call this function repeatedly to retrieve information about all columns in the result set.
maxdb_fetch_field() returns **`FALSE`** when no more fields are left.

Return values

Returns a resource which contains field definition information or **FALSE** if no field information is available.

Tabla 1. Object properties

Property	Description
name	The name of the column
max_length	The maximum width of the field for the result set.
type	The data type used for this field
decimals	The number of decimals used (for integer fields)

See also

maxdb_num_fields() maxdb_fetch_field_direct() maxdb_fetch_fields() maxdb_field_seek()

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

$query = "SELECT name, cno from hotel.customer ORDER BY cno";

if ($result = maxdb_query($link, $query)) {

   /* Get field information for all fields */
   while ($finfo = maxdb_fetch_field($result)) {

       printf("Name:     %s\n", $finfo->name);
       printf("Table:    %s\n", $finfo->table);
       printf("max. Len: %d\n", $finfo->max_length);
       printf("Flags:    %d\n", $finfo->flags);
       printf("Type:     %d\n\n", $finfo->type);
   }
   maxdb_free_result($result);
}

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
Name:      NAME
Table:
max. Len: 10
Flags:     -1
Type:      2

Name:      CNO
Table:
max. Len: 4
Flags:     -1
Type:      0
```

maxdb_fetch_fields

(no version information, might be only in CVS)

maxdb_fetch_fields -- Returns an array of resources representing the fields in a result set

Description

Procedural Style:

mixed **maxdb_fetch_fields** (resource result)

This function serves an identical purpose to the **maxdb_fetch_field()** function with the single difference that, instead of returning one resource at a time for each field, the columns are returned as an array of resources.

Return values

Returns an array of resources which contains field definition information or **FALSE** if no field information is available.

Tabla 1. Object properties

Property	Description
name	The name of the column
max_length	The maximum width of the field for the result set.
type	The data type used for this field
decimals	The number of decimals used (for integer fields)

See also

maxdb_num_fields() **maxdb_fetch_field()** **maxdb_fetch_field_direct()**

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

$query = "SELECT name, cno from hotel.customer ORDER BY cno";

if ($result = maxdb_query($link, $query)) {

   /* Get field information for all columns */
   $finfo = maxdb_fetch_fields($result);

   for ($i=0; $i < count($finfo); $i++) {
       printf("Name:     %s\n", $finfo[$i]->name);
       printf("Table:    %s\n", $finfo[$i]->table);
       printf("max. Len: %d\n", $finfo[$i]->max_length);
       printf("Flags:    %d\n", $finfo[$i]->flags);
       printf("Type:     %d\n\n", $finfo[$i]->type);
   }
   maxdb_free_result($result);
}

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
Name:     NAME
Table:
max. Len: 10
Flags:    -1
Type:     2

Name:     CNO
Table:
max. Len: 4
Flags:    -1
Type:     0
```

maxdb_fetch_lengths

(no version information, might be only in CVS)

maxdb_fetch_lengths -- Returns the lengths of the columns of the current row in the result set

Description

Procedural style:

mixed **maxdb_fetch_lengths** (resource result)

The **maxdb_fetch_lengths()** function returns an array containing the lengths of every column of the current row within the result set represented by the *result* parameter. If successful, a numerically indexed array representing the lengths of each column is returned or **FALSE** on failure.

Return values

An array of integers representing the size of each column (not including any terminating null characters). **FALSE** if an error occurred.

maxdb_fetch_lengths() is valid only for the current row of the result set. It returns **FALSE** if you call it before calling maxdb_fetch_row/array/resource or after retrieving all rows in the result.

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

$query = "SELECT * from hotel.customer WHERE cno = 3000";

if ($result = maxdb_query($link, $query)) {

   $row = maxdb_fetch_row($result);

   /* display column lengths */
   $lengths = maxdb_fetch_lengths($result);
   for ($i=0; $i < count($lengths); $i++) {
       printf("Field %2d has Length %2d\n", $i+1, $lengths[$i]);
   }
   maxdb_free_result($result);
}

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
Field  1 has Length  4
Field  2 has Length  3
Field  3 has Length  5
Field  4 has Length  6
Field  5 has Length  5
Field  6 has Length 21
```

maxdb_fetch_resource

(no version information, might be only in CVS)

maxdb_fetch_resource -- Returns the current row of a result set as an resource

Description

Procedural style:

mixed **maxdb_fetch_resource** (resource result)

The **maxdb_fetch_resource()** will return the current row result set as an resource where the attributes of the resource represent the names of the fields found within the result set. If no more rows exist in the current result set, **NULL** is returned.

Return values

Returns an resource that corresponds to the fetched row or **NULL** if there are no more rows in resultset.

> **Nota:** Los nombres de campos retornados por esta función diferencian entre *mayusculas y minusculas*.

> **Nota:** Esta funcion define campos NULL como valores PHP **NULL**.

See also

maxdb_fetch_array(), **maxdb_fetch_assoc()**, **maxdb_fetch_row()**.

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

$query = "SELECT name, state FROM hotel.city ORDER by zip";

if ($result = maxdb_query($link, $query)) {

   /* fetch associative array */
   while ($obj = maxdb_fetch_object($result)) {
       printf ("%s (%s)\n", $obj->NAME, $obj->STATE);
   }

   /* free result set */
   maxdb_free_result($result);
}

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
New York (NY)
New York (NY)
Long Island (NY)
Albany (NY)
Washington (DC)
Washington (DC)
Washington (DC)
Silver Spring (MD)
Daytona Beach (FL)
Deerfield Beach (FL)
Clearwater (FL)
Cincinnati (OH)
Detroit (MI)
Rosemont (IL)
Chicago (IL)
Chicago (IL)
New Orleans (LA)
Dallas (TX)
Los Angeles (CA)
Hollywood (CA)
Long Beach (CA)
Palm Springs (CA)
Irvine (CA)
Santa Clara (CA)
Portland (OR)
```

maxdb_fetch_row

(no version information, might be only in CVS)

maxdb_fetch_row -- Get a result row as an enumerated array

Description

Procedural style:

mixed **maxdb_fetch_row** (resource result)

Returns an array that corresponds to the fetched row, or **NULL** if there are no more rows.

maxdb_fetch_row() fetches one row of data from the result set represented by *result* and returns it as an enumerated array, where each column is stored in an array offset starting from 0 (zero). Each subsequent call to the **maxdb_fetch_row()** function will return the next row within the result set, or **FALSE** if there are no more rows.

Return values

maxdb_fetch_row() returns an array that corresponds to the fetched row or **NULL** if there are no more rows in result set.

> **Nota:** Esta funcion define campos NULL como valores PHP **NULL**.

See also

maxdb_fetch_array(), **maxdb_fetch_assoc()**, **maxdb_fetch_resource()**.

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

$query = "SELECT name, state FROM hotel.city ORDER by zip";

if ($result = maxdb_query($link, $query)) {

   /* fetch associative array */
   while ($row = maxdb_fetch_row($result)) {
       printf ("%s (%s)\n", $row[0], $row[1]);
   }

   /* free result set */
   maxdb_free_result($result);
}

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
New York (NY)
New York (NY)
Long Island (NY)
Albany (NY)
Washington (DC)
Washington (DC)
Washington (DC)
Silver Spring (MD)
Daytona Beach (FL)
Deerfield Beach (FL)
Clearwater (FL)
Cincinnati (OH)
Detroit (MI)
Rosemont (IL)
Chicago (IL)
Chicago (IL)
New Orleans (LA)
Dallas (TX)
Los Angeles (CA)
Hollywood (CA)
Long Beach (CA)
Palm Springs (CA)
Irvine (CA)
Santa Clara (CA)
Portland (OR)
```

maxdb_fetch

maxdb_fetch -- Alias for **maxdb_stmt_fetch()**

Description

This function is an alias of **maxdb_stmt_fetch()**. For a detailled descripton see description of

maxdb_stmt_fetch().

> **Nota: maxdb_fetch()** is deprecated and will be removed.

See also

maxdb_stmt_fetch()

maxdb_field_count

(no version information, might be only in CVS)

maxdb_field_count -- Returns the number of columns for the most recent query

Description

Procedural style:

int **maxdb_field_count** (resource link)

Returns the number of columns for the most recent query on the connection represented by the *link* parameter. This function can be useful when using the **maxdb_store_result()** function to determine if the query should have produced a non-empty result set or not without knowing the nature of the query.

Return values

An integer representing the number of fields in a result set

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

maxdb_report (MAXDB_REPORT_OFF);
maxdb_query($link,"DROP TABLE friends");
maxdb_report (MAXDB_REPORT_ERROR);

maxdb_query($link, "CREATE TABLE friends (id int, name varchar(20))");

maxdb_query($link, "INSERT INTO friends VALUES (1,'Hartmut')");
maxdb_query($link, "INSERT INTO friends VALUES (2, 'Ulf')");

if (maxdb_field_count($link)) {
   /* this was a select/show or describe query */
   $result = maxdb_store_result($link);

   /* process resultset */
   $row = maxdb_fetch_row($result);

   /* free resultset */
   maxdb_free_result($result);
}

/* close connection */
maxdb_close($link);
?>
```

maxdb_field_seek

(no version information, might be only in CVS)

maxdb_field_seek -- Set result pointer to a specified field offset

Description

Procedural style:

int **maxdb_field_seek** (resource result, int fieldnr)

Sets the field cursor to the given offset. The next call to **maxdb_fetch_field()** will retrieve the field definition of the column associated with that offset.

> **Nota:** To seek to the beginning of a row, pass an offset value of zero.

Return values

maxdb_field_seek() returns previuos value of field cursor.

See also

maxdb_fetch_field()

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

$query = "SELECT name, cno from hotel.customer ORDER BY cno";

if ($result = maxdb_query($link, $query)) {

   /* Get field information for 2nd column */
   maxdb_field_seek($result, 1);
   $finfo = maxdb_fetch_field($result);

   printf("Name:     %s\n", $finfo->name);
   printf("Table:    %s\n", $finfo->table);
   printf("max. Len: %d\n", $finfo->max_length);
   printf("Flags:    %d\n", $finfo->flags);
   printf("Type:     %d\n\n", $finfo->type);

   maxdb_free_result($result);
}

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
Name:     NAME
Table:
max. Len: 10
Flags:    -1
Type:     2
```

maxdb_field_tell

(no version information, might be only in CVS)

maxdb_field_tell -- Get current field offset of a result pointer

Description

Procedural style:

int **maxdb_field_tell** (resource result)

Returns the position of the field cursor used for the last **maxdb_fetch_field()** call. This value can be used as an argument to **maxdb_field_seek()**.

Valores retornados

Returns current offset of field cursor.

Ver también

maxdb_fetch_field(), **maxdb_field_seek()**

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

$query = "SELECT name, cno from hotel.customer ORDER BY cno";

if ($result = maxdb_query($link, $query)) {

   /* Get field information for all fields */
   while ($finfo = maxdb_fetch_field($result)) {

       /* get fieldpointer offset */
       $currentfield = maxdb_field_tell($result);

       printf("Column    %d:\n", $currentfield);
       printf("Name:     %s\n", $finfo->name);
       printf("Table:    %s\n", $finfo->table);
       printf("max. Len: %d\n", $finfo->max_length);
       printf("Flags:    %d\n", $finfo->flags);
       printf("Type:     %d\n\n", $finfo->type);
   }
   maxdb_free_result($result);
}

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
Column    1:
Name:     NAME
Table:
max. Len: 10
Flags:    -1
Type:     2

Column    2:
Name:     CNO
Table:
max. Len: 4
Flags:    -1
Type:     0
```

maxdb_free_result

(no version information, might be only in CVS)

maxdb_free_result -- Frees the memory associated with a result

Description

Procedural style:

void **maxdb_free_result** (resource result)

The **maxdb_free_result()** function frees the memory associated with the result represented by the *result* parameter, which was allocated by **maxdb_query()**, **maxdb_store_result()** or **maxdb_use_result()**.

> **Nota:** You should always free your result with **maxdb_free_result()**, when your result resource is not needed anymore.

Valores retornados

This function doesn't return any value.

Ver también

maxdb_query(), **maxdb_stmt_store_result()**, **maxdb_store_result()**, **maxdb_use_result()**.

maxdb_get_client_info

(no version information, might be only in CVS)

maxdb_get_client_info -- Returns the MaxDB client version as a string

Description

string **maxdb_get_client_info** (void)

The **maxdb_get_client_info()** function is used to return a string representing the client version being used in the MaxDB extension.

Valores retornados

A string that represents the MaxDB client library version

See also

maxdb_get_client_version(), **maxdb_get_server_info()**, **maxdb_get_server_version()**

Example

Ejemplo 1. maxdb_get_client_info

```
<?php

/* We don't need a connection to determine
   the version of MaxDB client library */

printf("Client library version: %s\n", maxdb_get_client_info());
?>
```

The above examples would produce the following output:

```
Client library version: libSQLDBC <...>
```

maxdb_get_client_version

(no version information, might be only in CVS)

maxdb_get_client_version -- Get MaxDB client info

Description

int **maxdb_get_client_version** (void)

Returns client version number as an integer.

Return values

A number that represents the MaxDB client library version in format: *main_version*10000* + *minor_version *100* + *sub_version*. For example, 7.5.0 is returned as 70500.

This is useful to quickly determine the version of the client library to know if some capability exits.

See also

maxdb_get_client_info(), **maxdb_get_server_info()**, **maxdb_get_server_version()**

Example

Ejemplo 1. maxdb_get_client_version

```
<?php

/* We don't need a connection to determine
   the version of MaxDB client library */

printf("Client library version: %d\n", maxdb_get_client_version());
?>
```

The above examples would produce the following output:

```
Client library version: 7.<...>
```

maxdb_get_host_info

(no version information, might be only in CVS)

maxdb_get_host_info -- Returns a string representing the type of connection used

Description

Procdural style:

string **maxdb_get_host_info** (resource link)

The **maxdb_get_host_info()** function returns a string describing the connection represented by the

link parameter is using.

Valores retornados

A character string representing the server hostname and the connection type.

See also

maxdb_get_proto_info()

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

/* print host information */
printf("Host info: %s\n", maxdb_get_host_info($link));

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
Host info: localhost
```

maxdb_get_metadata

maxdb_get_metadata -- Alias for **maxdb_stmt_result_metadata()**

Description

This function is an alias of **maxdb_stmt_result_metadata()**. For a detailled descripton see description of **maxdb_stmt_result_metadata()**.

> **Nota: maxdb_get_metadata()** is deprecated and will be removed.

See also

maxdb_stmt_result_metadata()

maxdb_get_proto_info

(no version information, might be only in CVS)

maxdb_get_proto_info -- Returns the version of the MaxDB protocol used

Description

Procedural style:

int **maxdb_get_proto_info** (resource link)

Returns an integer representing the MaxDB protocol version used by the connection represented by the *link* parameter.

Valores retornados

Returns an integer representing the protocol version (constant 10).

See also

maxdb_get_host_info()

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

/* print protocol version */
printf("Protocol version: %d\n", maxdb_get_proto_info($link));

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
Protocol version: 10
```

maxdb_get_server_info

(no version information, might be only in CVS)

maxdb_get_server_info -- Returns the version of the MaxDB server

Description

Procedural style:

string **maxdb_get_server_info** (resource link)

Returns a string representing the version of the MaxDB server that the MaxDB extension is connected to (represented by the *link* parameter).

Valores retornados

A character string representing the server version.

See also

maxdb_get_client_info(), **maxdb_get_client_version()**, **maxdb_get_server_version()**

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

/* print server version */
printf("Server version: %s\n", maxdb_get_server_info($link));

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
Server version: Kernel    7<...>
```

maxdb_get_server_version

(no version information, might be only in CVS)

maxdb_get_server_version -- Returns the version of the MaxDB server as an integer

Description

Procedural style:

int **maxdb_get_server_version** (resource link)

The **maxdb_get_server_version()** function returns the version of the server connected to (represented by the *link* parameter) as an integer.

The form of this version number is *main_version * 10000 + minor_version * 100 + sub_version* (i.e. version 7.5.0 is 70500).

Valores retornados

An integer representing the server version.

See also

maxdb_get_client_info(), maxdb_get_client_version(), maxdb_get_server_info()

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

/* print server version */
printf("Server version: %d\n", maxdb_get_server_version($link));

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
Server version: 7<...>
```

maxdb_info

(no version information, might be only in CVS)

maxdb_info -- Retrieves information about the most recently executed query

Description

Procedural style:

string **maxdb_info** (resource link)

The **maxdb_info()** function returns a string providing information about the last query executed. The nature of this string is provided below:

Tabla 1. Possible maxdb_info return values

Query type	Example result string
INSERT INTO...SELECT...	Records: 100 Duplicates: 0 Warnings: 0
INSERT INTO...VALUES (...),(...),(...)	Records: 3 Duplicates: 0 Warnings: 0
LOAD DATA INFILE ...	Records: 1 Deleted: 0 Skipped: 0 Warnings: 0
ALTER TABLE ...	Records: 3 Duplicates: 0 Warnings: 0
UPDATE ...	Rows matched: 40 Changed: 40 Warnings: 0

Nota: Queries which do not fall into one of the above formats are not supported. In

these situations, **maxdb_info()** will return an empty string.

Valores retornados

A character string representing additional information about the most recently executed query.

Ver también

maxdb_affected_rows(), **maxdb_warning_count()**, **maxdb_num_rows()**

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

maxdb_query($link, "CREATE TABLE temp.t1 LIKE hotel.city");

/* INSERT INTO .. SELECT */
maxdb_query($link, "INSERT INTO temp.t1 SELECT * FROM hotel.city");
printf("%s\n", maxdb_info($link));

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
Records: 25 Duplicates: 0 Warnings: 0
```

maxdb_init

(no version information, might be only in CVS)

maxdb_init -- Initializes MaxDB and returns an resource for use with maxdb_real_connect

Description

resource **maxdb_init** (void)

Allocates or initializes a MaxDB resource suitable for **maxdb_options()** and **maxdb_real_connect()**.

> **Nota:** Any subsequent calls to any maxdb function (except **maxdb_options()**) will fail until **maxdb_real_connect()** was called.

Valores retornados

Returns an resource.

Ver también

maxdb_options(), **maxdb_close()**, **maxdb_real_connect()**, **maxdb_connect()**

maxdb_insert_id

(no version information, might be only in CVS)

maxdb_insert_id -- Returns the auto generated id used in the last query

Description

Procedural style:

mixed **maxdb_insert_id** (resource link)

The **maxdb_insert_id()** function returns the ID generated by a query on a table with a column having the DEFAULT SERIAL attribute. If the last query wasn't an INSERT or UPDATE statement or if the modified table does not have a column with the DEFAULT SERIAL attribute, this function will return zero.

Valores retornados

The value of the *DEFAULT SERIAL* field that was updated by the previous query. Returns zero if there was no previous query on the connection or if the query did not update an *DEFAULT_SERIAL* value.

> **Nota:** If the number is greater than maximal int value, **maxdb_insert_id()** will return a string.

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

maxdb_report (MAXDB_REPORT_OFF);
maxdb_query($link,"DROP TABLE mycity");
maxdb_report (MAXDB_REPORT_ERROR);

maxdb_query($link, "CREATE TABLE mycity LIKE hotel.city");
maxdb_query($link, "ALTER TABLE mycity ADD id FIXED(11) DEFAULT SERIAL");

$query = "INSERT INTO mycity (zip,name,state) VALUES ('12203','Albany' ,'NY')";
maxdb_query($link, $query);

printf ("New Record has id %d.\n", maxdb_insert_id($link));

/* drop table */
maxdb_query($link, "DROP TABLE mycity");

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
New Record has id 1.
```

maxdb_kill

(no version information, might be only in CVS)

maxdb_kill -- Disconnects from a MaxDB server

Description

Procedural style:

bool **maxdb_kill** (resource link, int processid)

This function is used to disconnect from a MaxDB server specified by the *processid* parameter.

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Ver también

maxdb_thread_id()

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

/* determine our thread id */
$thread_id = maxdb_thread_id($link);

/* Kill connection */
maxdb_kill($link, $thread_id);

/* This should produce an error */
if (!maxdb_query($link, "CREATE TABLE myCity LIKE City")) {
   printf("Error: %s\n", maxdb_error($link));
   exit;
}

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
Error: Session not connected
```

maxdb_master_query

(no version information, might be only in CVS)

maxdb_master_query -- Enforce execution of a query on the master in a master/slave setup

Description

bool **maxdb_master_query** (resource link, string query)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

maxdb_more_results

(no version information, might be only in CVS)

maxdb_more_results -- Check if there any more query results from a multi query

Description

bool **maxdb_more_results** (resource link)

maxdb_more_results() indicates if one or more result sets are available from a previous call to

maxdb_multi_query().

Return values

Always **FALSE**.

See also

maxdb_multi_query(), maxdb_next_result(), maxdb_store_result(), maxdb_use_result()

Example

See maxdb_multi_query().

maxdb_multi_query

(no version information, might be only in CVS)

maxdb_multi_query -- Performs a query on the database

Description

Procedural style:

bool **maxdb_multi_query** (resource link, string query)

The **maxdb_multi_query()** works like the function maxdb_query(). Multiple queries are not yet supported.

Return values

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

See also

maxdb_use_result(), maxdb_store_result(), maxdb_next_result(), maxdb_more_results()

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

$query = "SELECT * FROM dual";
$query .= "SELECT name FROM hotel.city ORDER BY zip";

/* execute multi query */
if (maxdb_multi_query($link, $query)) {
   do {
       /* store first result set */
       if ($result = maxdb_store_result($link)) {
           while ($row = maxdb_fetch_row($result)) {
               printf("%s\n", $row[0]);
           }
           maxdb_free_result($result);
       }
       /* print divider */
       if (maxdb_more_results($link)) {
           printf("-----------------\n");
       }
   } while (maxdb_next_result($link));
}

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
Warning: maxdb_multi_query(): -3008 POS(31) Invalid keyword or missing delimiter <...>
```

maxdb_next_result

(no version information, might be only in CVS)

maxdb_next_result -- Prepare next result from multi_query

Description

bool **maxdb_next_result** (resource link)

Since multiple queries are not yet supported, **maxdb_next_result()** returns always **FALSE**.

Return values

FALSE

See also

maxdb_multi_query(), **maxdb_more_results()**, **maxdb_store_result()**, **maxdb_use_result()**

maxdb_num_fields

(no version information, might be only in CVS)

maxdb_num_fields -- Get the number of fields in a result

Description

Procedural style:

int **maxdb_num_fields** (resource result)

maxdb_num_fields() returns the number of fields from specified result set.

Valores retornados

The number of fields from a result set

Ver también

maxdb_fetch_field()

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

if ($result = maxdb_query($link, "SELECT * FROM hotel.city ORDER BY zip")) {

   /* determine number of fields in result set */
   $field_cnt = maxdb_num_fields($result);

   printf("Result set has %d fields.\n", $field_cnt);

   /* close result set */
   maxdb_free_result($result);
}

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
Result set has 3 fields.
```

maxdb_num_rows

(no version information, might be only in CVS)

maxdb_num_rows -- Gets the number of rows in a result

Description

Procedural style:

mixed **maxdb_num_rows** (resource result)

Returns the number of rows in the result set.

The use of **maxdb_num_rows()** depends on whether you use buffered or unbuffered result sets. In case you use unbuffered resultsets **maxdb_num_rows()** will not correct the correct number of rows until all the rows in the result have been retrieved.

Valores retornados

Returns number of rows in the result set.

> **Nota:** If the number of rows is greater than maximal int value, the number will be returned as a string.

Ver también

maxdb_affected_rows() **maxdb_store_result()** **maxdb_use_result()** **maxdb_query()**

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

if ($result = maxdb_query($link, "SELECT cno, name FROM hotel.customer ORDER BY name"))

   /* determine number of rows result set */
   $row_cnt = maxdb_num_rows($result);

   printf("Result set has %d rows.\n", $row_cnt);

   /* close result set */
   maxdb_free_result($result);
}

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
Result set has -1 rows.
```

maxdb_options

(no version information, might be only in CVS)

maxdb_options -- Set options

Description

Procedural style:

bool **maxdb_options** (resource link, int option, mixed value)

maxdb_options() can be used to set extra connect options and affect behavior for a connection.

This function may be called multiple times to set several options.

maxdb_options() should be called after **maxdb_init()** and before **maxdb_real_connect()**.

The parameter *option* is the option that you want to set, the *value* is the value for the option. For detailed description of the options see http://dev.mysql.com/doc/maxdb/ The parameter *option* can be one of the following values:

Tabla 1. Valid options

Name	Description
MAXDB_COMPNAME	The component name used to initialise the SQLDBC runtime environment.
MAXDB_APPLICATION	The application to be connected to the database.
MAXDB_APPVERSION	The version of the application.
MAXDB_SQLMODE	The SQL mode.
MAXDB_UNICODE	TRUE, if the connection is an unicode (UCS2) client or FALSE, if not.
MAXDB_TIMEOUT	The maximum allowed time of inactivity after which the connection to the database is closed by the system.
MAXDB_ISOLATIONLEVEL	Specifies whether and how shared locks and exclusive locks are implicitly requested or released.
MAXDB_PACKETCOUNT	The number of different request packets used for the connection.
MAXDB_STATEMENTCACHESIZE	The number of prepared statements to be cached for the connection for re-use.
MAXDB_CURSORPREFIX	The prefix to use for result tables that are automatically named.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ver también

maxdb_init(), **maxdb_real_connect()**

Example

See **maxdb_real_connect()**.

maxdb_param_count

maxdb_param_count -- Alias for **maxdb_stmt_param_count()**

Description

This function is an alias of **maxdb_stmt_param_count()**. For a detailled descripton see description of **maxdb_stmt_param_count()**.

> **Nota: maxdb_param_count()** is deprecated and will be removed.

See also

maxdb_stmt_param_count()

maxdb_ping

(no version information, might be only in CVS)

maxdb_ping -- Pings a server connection, or tries to reconnect if the connection has gone down

Description

Procedural style:

bool **maxdb_ping** (resource link)

Checks whether the connection to the server is working. If it has gone down, and global option *maxdb.reconnect* is enabled an automatic reconnection is attempted.

This function can be used by clients that remain idle for a long while, to check whether the server has closed the connection and reconnect if necessary.

Return values

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

/* check if server is alive */
if (maxdb_ping($link)) {
   printf ("Our connection is ok!\n");
} else {
   printf ("Error: %s\n", maxdb_error($link));
}

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
Our connection is ok!
```

maxdb_prepare

(no version information, might be only in CVS)

maxdb_prepare -- Prepare a SQL statement for execution

Description

Procedure style:

mixed **maxdb_prepare** (resource link, string query)

maxdb_prepare() prepares the SQL query pointed to by the null-terminated string query, and returns a statement handle to be used for further operations on the statement. The query must consist of a single SQL statement.

> **Nota:** You should not add a terminating semicolon or *g* to the statement.

The parameter *query* can include one or more parameter markers in the SQL statement by embedding question mark (*?*) characters at the appropriate positions.

> **Nota:** The markers are legal only in certain places in SQL statements. For example, they are allowed in the VALUES() list of an INSERT statement (to specify column values for a row), or in a comparison with a column in a WHERE clause to specify a comparison value.
>
> However, they are not allowed for identifiers (such as table or column names), in the select list that names the columns to be returned by a SELECT statement), or to specify both operands of a binary operator such as the = equal sign. The latter restriction is necessary because it would be impossible to determine the parameter type. In general, parameters are legal only in Data Manipulation Languange (DML) statements, and not in Data Defination Language (DDL) statements.

The parameter markers must be bound to application variables using **maxdb_stmt_bind_param()** and/or **maxdb_stmt_bind_result()** before executing the statement or fetching rows.

Valores retornados

maxdb_prepare() returns a statement resource or **FALSE** if an error occured.

Ver también

maxdb_stmt_execute(), **maxdb_stmt_fetch()**, **maxdb_stmt_bind_param()**, **maxdb_stmt_bind_result()**, **maxdb_stmt_close()**

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

$city = "Rosemont";

/* create a prepared statement */
if ($stmt = maxdb_prepare($link, "SELECT state FROM hotel.city WHERE name=?")) {

   /* bind parameters for markers */
   maxdb_stmt_bind_param($stmt, "s", $city);

   /* execute query */
   maxdb_stmt_execute($stmt);

  /* bind result variables */
   maxdb_stmt_bind_result($stmt, $district);

   /* fetch value */
   maxdb_stmt_fetch($stmt);

   printf("%s is in district %s\n", $city, $district);

   /* close statement */
   maxdb_stmt_close($stmt);
}

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
Rosemont is in district IL
```

maxdb_query

(no version information, might be only in CVS)

maxdb_query -- Performs a query on the database

Description

Procedural style:

mixed **maxdb_query** (resource link, string query [, int resultmode])

The **maxdb_query()** function is used to simplify the act of performing a query against the database represented by the *link* parameter.

Return values

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo. For *SELECT, SHOW, DESCRIBE* or *EXPLAIN* **maxdb_query()** will return a result resource.

See also

maxdb_real_query(), **maxdb_multi_query()**, **maxdb_free_result()**

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

/* Create table doesn't return a resultset */
if (maxdb_query($link, "CREATE TABLE temp.mycity LIKE hotel.city") === TRUE) {
   printf("Table mycity successfully created.\n");
}

/* Select queries return a resultset */
if ($result = maxdb_query($link, "SELECT name FROM hotel.city")) {
   printf("Select returned %d rows.\n", maxdb_num_rows($result));

   /* free result set */
   maxdb_free_result($result);
}

/* If we have to retrieve large amount of data we use MAXDB_USE_RESULT */
if ($result = maxdb_query($link, "SELECT * FROM hotel.city", MAXDB_USE_RESULT)) {
   maxdb_free_result($result);
}

maxdb_close($link);
?>
```

The above examples would produce the following output:

```
Table mycity successfully created.
Select returned -1 rows.
```

maxdb_real_connect

(no version information, might be only in CVS)

maxdb_real_connect -- Opens a connection to a MaxDB server

Description

Procedural style

bool **maxdb_real_connect** (resource link [, string hostname [, string username [, string passwd [, string dbname [, int port [, string socket]]]]]])

maxdb_real_connect() attempts to establish a connection to a MaxDB database engine running on *hostname*.

This function differs from **maxdb_connect()**:

- **maxdb_real_connect()** needs a valid resource which has to be created by function **maxdb_init()**
- With function **maxdb_options()** you can set various options for connection.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ver también

maxdb_connect(), **maxdb_init()**, **maxdb_options()**, **maxdb_ssl_set()**, **maxdb_close()**.

Example

Ejemplo 1. Procedural style

```
<?php

/* create a connection object which is not connected */
$link = maxdb_init();

/* set connection options */
maxdb_options($link, MAXDB_UNICODE, "FALSE");
maxdb_options($link, MAXDB_TIMEOUT, 5);

/* connect to server */
maxdb_real_connect($link, 'localhost', 'MONA', 'RED');

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

printf ("Connection: %s\n.", maxdb_get_host_info($link));

maxdb_close($link);
?>
```

The above examples would produce the following output:

```
Connection: localhost <...>
```

maxdb_real_escape_string

(no version information, might be only in CVS)

maxdb_real_escape_string -- Escapes special characters in a string for use in a SQL statement, taking into account the current charset of the connection

Description

Procedural style:

string **maxdb_real_escape_string** (resource link, string escapestr)

This function is used to create a legal SQL string that you can use in a SQL statement. The string *escapestr* is encoded to an escaped SQL string, taking into account the current character set of the connection.

Characters encoded are ′, ″.

Return values

Returns an escaped string.

See also

maxdb_character_set_name().

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

maxdb_query($link, "CREATE TABLE temp.mycity LIKE hotel.city");

$city = "'s Hertogenbosch";

/* this query will fail, cause we didn't escape $city */
if (!maxdb_query($link, "INSERT into temp.mycity VALUES ('11111','$city','NY')")) {
   printf("Error: %s\n", maxdb_sqlstate($link));
}

$city = maxdb_real_escape_string($link, $city);

/* this query with escaped $city will work */
if (maxdb_query($link, "INSERT into temp.mycity VALUES ('22222','$city','NY')")) {
   printf("%d Row inserted.\n", maxdb_affected_rows($link));
}

maxdb_close($link);
?>
```

The above examples would produce the following output:

```
Warning: maxdb_query(): -5016 POS(43) Missing delimiter: ) <...>
Error: 42000
1 Row inserted.
```

maxdb_real_query

(no version information, might be only in CVS)

maxdb_real_query -- Execute an SQL query

Description

Procedural style

bool **maxdb_real_query** (resource link, string query)

The **maxdb_real_query()** is functionally identical with the **maxdb_query()**.

> **Nota:** In order to determine if a given query should return a result set or not, see **maxdb_field_count()**.

Return values

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

See also

maxdb_query() **maxdb_store_result()** **maxdb_use_result()**

maxdb_report

(no version information, might be only in CVS)

maxdb_report -- Enables or disables internal report functions

Description

bool **maxdb_report** (int flags)

Example

Ejemplo 1. Procedural style

```
<?php
/* activate reporting */
maxdb_report(MAXDB_REPORT_ERROR);

$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

/* this query should report an error */
$result = maxdb_query($link,"SELECT Name FROM Nonexistingtable WHERE population > 50000

maxdb_close($link);
?>
```

The above examples would produce the following output:

```
Warning: maxdb_query(): -4004 POS(18) Unknown table name:NONEXISTINGTABLE <...>
```

maxdb_rollback

(no version information, might be only in CVS)

maxdb_rollback -- Rolls back current transaction

Description

bool **maxdb_rollback** (resource link)

Rollbacks the current transaction for the database specified by the *link* parameter.

Return values

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

See also

maxdb_commit() maxdb_autocommit()

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

/* disable autocommit */
maxdb_autocommit($link, FALSE);

maxdb_query($link, "CREATE TABLE temp.mycity LIKE hotel.city");
maxdb_query($link, "INSERT INTO temp.mycity SELECT * FROM hotel.city");

/* commit insert */
maxdb_commit($link);

/* delete all rows */
maxdb_query($link, "DELETE FROM temp.mycity");

if ($result = maxdb_query($link, "SELECT COUNT(*) FROM temp.mycity")) {
   $row = maxdb_fetch_row($result);
   printf("%d rows in table mycity.\n", $row[0]);
   /* Free result */
   maxdb_free_result($result);
}

/* Rollback */
maxdb_rollback($link);

if ($result = maxdb_query($link, "SELECT COUNT(*) FROM temp.mycity")) {
   $row = maxdb_fetch_row($result);
   printf("%d rows in table mycity (after rollback).\n", $row[0]);
   /* Free result */
   maxdb_free_result($result);
}

/* Drop table myCity */
maxdb_query($link, "DROP TABLE temp.mycity");

maxdb_close($link);
?>
```

The above examples would produce the following output:

```
0 rows in table mycity.
25 rows in table mycity (after rollback).
```

maxdb_rpl_parse_enabled

(no version information, might be only in CVS)

maxdb_rpl_parse_enabled -- Check if RPL parse is enabled

Description

int **maxdb_rpl_parse_enabled** (resource link)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

maxdb_rpl_probe

(no version information, might be only in CVS)

maxdb_rpl_probe -- RPL probe

Description

bool **maxdb_rpl_probe** (resource link)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

maxdb_rpl_query_type

(no version information, might be only in CVS)

maxdb_rpl_query_type -- Returns RPL query type

Description

int **maxdb_rpl_query_type** (resource link, string query)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

maxdb_select_db

(no version information, might be only in CVS)

maxdb_select_db -- Selects the default database for database queries

Description

bool **maxdb_select_db** (resource link, string dbname)

The **maxdb_select_db()** function selects the default database (specified by the *dbname* parameter) to be used when performing queries against the database connection represented by the *link* parameter.

> **Nota:** This function should only be used to change the default database for the connection. You can select the default database with 4th parameter in **maxdb_connect ()**.

Return values

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

See also

maxdb_connect() **maxdb_real_connect()**

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

/* return name of current default database */
if ($result = maxdb_query($link, "SELECT SERVERDB FROM USERS WHERE USERNAME='MONA'")) {
   $row = maxdb_fetch_row($result);
   printf("Default database is %s.\n", $row[0]);
   maxdb_free_result($result);
}

/* change db to world db */
maxdb_select_db($link, "XXXXXXXX");

/* return name of current default database */
if ($result = maxdb_query($link, "SELECT SERVERDB FROM USERS WHERE USERNAME='MONA'")) {
   $row = maxdb_fetch_row($result);
   printf("Default database is %s.\n", $row[0]);
   maxdb_free_result($result);
}

maxdb_close($link);
?>
```

The above examples would produce the following output:

```
Default database is DB76.

Warning: maxdb_select_db(): -10709 Connection failed (RTE:database not running) <...>

Warning: maxdb_query(): -10821 Session not connected [] <...>

Warning: maxdb_close(): -10821 Session not connected [] <...>
```

maxdb_send_long_data

maxdb_send_long_data -- Alias for **maxdb_stmt_send_long_data()**

Description

This function is an alias of **maxdb_stmt_send_long_data()**. For a detailled descripton see description of **maxdb_stmt_send_long_data()**.

> **Nota: maxdb_send_long_data()** is deprecated and will be removed.

See also

maxdb_stmt_send_long_data()

maxdb_send_query

(no version information, might be only in CVS)

maxdb_send_query -- Send the query and return

Description

bool **maxdb_send_query** (resource link, string query)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

maxdb_server_end

(no version information, might be only in CVS)

maxdb_server_end -- Shut down the embedded server

Description

void **maxdb_server_end** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

maxdb_server_init

(no version information, might be only in CVS)

maxdb_server_init -- Initialize embedded server

Description

bool **maxdb_server_init** ([array server [, array groups]])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

maxdb_set_opt

maxdb_set_opt -- Alias of **maxdb_options()**

Description

This function is an alias of **maxdb_options()**.

maxdb_sqlstate

(no version information, might be only in CVS)

maxdb_sqlstate -- Returns the SQLSTATE error from previous MaxDB operation

Description

Procedural style:

string **maxdb_sqlstate** (resource link)

Returns a string containing the SQLSTATE error code for the last error. The error code consists of five characters. *'00000'* means no error. The values are specified by ANSI SQL and ODBC.

> **Nota:** Note that not all MaxDB errors are yet mapped to SQLSTATE's. The value *HY000* (general error) is used for unmapped errors.

Return values

Returns a string containing the SQLSTATE error code for the last error. The error code consists of five characters. *'00000'* means no error.

See also

maxdb_errno(), **maxdb_error()**

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

/* Table City already exists, so we should get an error */
if (!maxdb_query($link, "CREATE TABLE hotel.city (ID INT, Name VARCHAR(30))")) {
   printf("Error - SQLSTATE %s.\n", maxdb_sqlstate($link));
}

maxdb_close($link);
?>
```

The above examples would produce the following output:

```
Warning: maxdb_query(): -6000 POS(20) Duplicate table name:CITY [I6000] <...>
Error - SQLSTATE I6000.
```

maxdb_ssl_set

(no version information, might be only in CVS)

maxdb_ssl_set -- Used for establishing secure connections using SSL

Description

Procedural style:

bool **maxdb_ssl_set** (resource link, string key, string cert, string ca, string capath, string cipher)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

maxdb_stat

(no version information, might be only in CVS)

maxdb_stat -- Gets the current system status

Description

Procedural style:

mixed **maxdb_stat** (resource link)

maxdb_stat() returns a string containing several information about the MaxDB server running.

Return values

A string describing the server status. **FALSE** if an error occurred.

See also

maxdb_get_server_info()

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

printf("System status: %s\n", maxdb_stat($link));

maxdb_close($link);
?>
```

The above examples would produce the following output:

```
System status: Kernel    7<...>
```

maxdb_stmt_affected_rows

(no version information, might be only in CVS)

maxdb_stmt_affected_rows -- Returns the total number of rows changed, deleted, or inserted by the last executed statement

Description

Procedural style :

mixed **maxdb_stmt_affected_rows** (resource stmt)

maxdb_stmt_affected_rows() returns the number of rows affected by INSERT, UPDATE, or DELETE query. If the last query was invalid or the number of rows can not determined, this function will return -1.

Return Values

An integer greater than zero indicates the number of rows affected or retrieved. Zero indicates that no records where updated for an UPDATE/DELETE statement, no rows matched the WHERE clause in the query or that no query has yet been executed. -1 indicates that the query has returned an error or the number of rows can not determined.

See also

maxdb_stmt_num_rows(), **maxdb_prepare()**

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

/* create temp table */
maxdb_query($link, "CREATE TABLE temp.mycity LIKE hotel.city");

$query = "INSERT INTO temp.mycity SELECT * FROM hotel.city WHERE state LIKE ?";

/* prepare statement */
if ($stmt = maxdb_prepare($link, $query)) {

   /* Bind variable for placeholder */
   $code = 'N%';
   maxdb_stmt_bind_param($stmt, "s", $code);

   /* execute statement */
   maxdb_stmt_execute($stmt);

   printf("rows inserted: %d\n", maxdb_stmt_affected_rows($stmt));

   /* close statement */
   maxdb_stmt_close($stmt);
}

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
rows inserted: 4
```

maxdb_stmt_bind_param

(no version information, might be only in CVS)

maxdb_stmt_bind_param -- Binds variables to a prepared statement as parameters

Description

Procedural style:

bool **maxdb_stmt_bind_param** (resource stmt, string types, mixed &var1 [, mixed &...])

maxdb_stmt_bind_param() is used to bind variables for the parameter markers in the SQL statement that was passed to **maxdb_prepare()**. The string *types* contains one or more characters which specify the types for the corresponding bind variables

Tabla 1. Type specification chars

Character	Description
i	corresponding variable has type integer
d	corresponding variable has type double
s	corresponding variable has type string
b	corresponding variable is a blob and will be send in packages

Return values

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

See also

maxdb_stmt_bind_result(), **maxdb_stmt_execute()**, **maxdb_stmt_fetch()**, **maxdb_prepare()**, **maxdb_stmt_send_long_data()**, **maxdb_stmt_errno()**, **maxdb_stmt_error()**

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (!$link) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

maxdb_query ($link, "CREATE TABLE temp.mycity LIKE hotel.city");
maxdb_query ($link, "INSERT INTO temp.mycity SELECT * FROM hotel.city");

$stmt = maxdb_prepare($link, "INSERT INTO temp.mycity VALUES (?, ?, ?)");
maxdb_stmt_bind_param($stmt, 'sss', $zip, $name, $state);

$zip = '11111';
$name = 'Georgetown';
$state = 'NY';

/* execute prepared statement */
maxdb_stmt_execute($stmt);

printf("%d Row inserted.\n", maxdb_stmt_affected_rows($stmt));

/* close statement and connection */
maxdb_stmt_close($stmt);

/* Clean up table CountryLanguage */
maxdb_query($link, "DELETE FROM temp.mycity WHERE name='Georgetown'");
printf("%d Row deleted.\n", maxdb_affected_rows($link));

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
1 Row inserted.
1 Row deleted.
```

maxdb_stmt_bind_result

(no version information, might be only in CVS)

maxdb_stmt_bind_result -- Binds variables to a prepared statement for result storage

Description

Procedural style:

bool **maxdb_stmt_bind_result** (resource stmt, mixed &var1 [, mixed &...])

maxdb_stmt_bind_result() is used to associate (bind) columns in the result set to variables. When **maxdb_stmt_fetch()** is called to fetch data, the MaxDB client/server protocol places the data for the bound columns into the specified variables *var1, ...*

> **Nota:** Note that all columns must be bound prior to calling **maxdb_stmt_fetch()**. Depending on column types bound variables can silently change to the corresponding PHP type.
>
> A column can be bound or rebound at any time, even after a result set has been partially

retrieved. The new binding takes effect the next time **maxdb_stmt_fetch()** is called.

Return values

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

See also

maxdb_stmt_bind_param(), **maxdb_stmt_execute()**, **maxdb_stmt_fetch()**, **maxdb_prepare()**, **maxdb_stmt_prepare()**, **maxdb_stmt_init()**, **maxdb_stmt_errno()**, **maxdb_stmt_error()**

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (!$link) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

/* prepare statement */
if ($stmt = maxdb_prepare($link, "SELECT zip, name FROM hotel.city ORDER BY name")) {
   maxdb_stmt_execute($stmt);

   /* bind variables to prepared statement */
   maxdb_stmt_bind_result($stmt, $col1, $col2);

   /* fetch values */
   while (maxdb_stmt_fetch($stmt)) {
       printf("%s %s\n", $col1, $col2);
   }

   /* close statement */
   maxdb_stmt_close($stmt);
}

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
12203 Albany
60601 Chicago
60615 Chicago
45211 Cincinnati
33575 Clearwater
75243 Dallas
32018 Daytona Beach
33441 Deerfield Beach
48226 Detroit
90029 Hollywood
92714 Irvine
90804 Long Beach
11788 Long Island
90018 Los Angeles
70112 New Orleans
10019 New York
10580 New York
92262 Palm Springs
97213 Portland
60018 Rosemont
95054 Santa Clara
20903 Silver Spring
20005 Washington
20019 Washington
20037 Washington
```

maxdb_stmt_close_long_data

(no version information, might be only in CVS)

maxdb_stmt_close_long_data -- Ends a sequence of **maxdb_stmt_send_long_data()**

Description

Procedural style:

bool **maxdb_stmt_close_long_data** (resource stmt, int param_nr)

This function has to be called after a sequence of **maxdb_stmt_send_long_data()**, that was started after **maxdb_execute()**.

param_nr indicates which parameter to associate the end of data with. Parameters are numbered beginning with 0.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ver también

maxdb_prepare(), **maxdb_stmt_bind_param()**

maxdb_stmt_close

(no version information, might be only in CVS)

maxdb_stmt_close -- Closes a prepared statement

Description

Procedural style:

bool **maxdb_stmt_close** (resource stmt)

Closes a prepared statement. **maxdb_stmt_close()** also deallocates the statement handle pointed to by *stmt*. If the current statement has pending or unread results, this function cancels them so that the next query can be executed.

Return values

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

See also

maxdb_prepare(),

maxdb_stmt_data_seek

(no version information, might be only in CVS)

maxdb_stmt_data_seek -- Seeks to an arbitray row in statement result set

Description

Procedural style:

bool **maxdb_stmt_data_seek** (resource statement, int offset)

The **maxdb_stmt_data_seek()** function seeks to an arbitrary result pointer specified by the *offset* in the statement result set represented by *statement*. The *offset* parameter must be between zero and the total number of rows minus one (0..**maxdb_stmt_num_rows()** - 1).

Return values

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

See also

maxdb_prepare()

Example

Ejemplo 1. Procedural style

```
<?php
/* Open a connection */
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

$query = "SELECT name, zip FROM hotel.city ORDER BY name";
if ($stmt = maxdb_prepare($link, $query)) {

   /* execute query */
   maxdb_stmt_execute($stmt);

   /* bind result variables */
   maxdb_stmt_bind_result($stmt, $name, $code);

   /* store result */
   maxdb_stmt_store_result($stmt);

   /* seek to row no. 400 */
   maxdb_stmt_data_seek($stmt, 5);

   /* fetch values */
   maxdb_stmt_fetch($stmt);

   printf ("City: %s  Zip: %s\n", $name, $code);

   /* close statement */
   maxdb_stmt_close($stmt);
}

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
City: Dallas  Zip: 75243
```

maxdb_stmt_errno

(no version information, might be only in CVS)

maxdb_stmt_errno -- Returns the error code for the most recent statement call

Description

Procedural style :

int **maxdb_stmt_errno** (resource stmt)

For the statement specified by *stmt*, **maxdb_stmt_errno()** returns the error code for the most recently invoked statement function that can succeed or fail.

> **Nota:** For possible error codes see documentation of SQLDBC: http://dev.mysql.com/doc/maxdb/.

Return values

An error code value. Zero means no error occurred.

See also

maxdb_stmt_error(), **maxdb_stmt_sqlstate()**

Example

Ejemplo 1. Procedural style

```
<?php
/* Open a connection */
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

maxdb_query($link, "CREATE TABLE temp.mycity LIKE hotel.city");
maxdb_query($link, "INSERT INTO temp.mycity SELECT * FROM hotel.city");


$query = "SELECT name, zip FROM temp.mycity ORDER BY name";
if ($stmt = maxdb_prepare($link, $query)) {

   /* drop table */
   maxdb_query($link, "DROP TABLE temp.mycity");

   /* execute query */
   maxdb_stmt_execute($stmt);

   printf("Error: %d.\n", maxdb_stmt_errno($stmt));

   /* close statement */
   maxdb_stmt_close($stmt);
}

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
Warning: maxdb_stmt_execute(): -4004 POS(23) Unknown table name:MYCITY [42000] <...>
Error: -4004.
```

maxdb_stmt_error

(no version information, might be only in CVS)

maxdb_stmt_error -- Returns a string description for last statement error

Description

Procedural style:

string **maxdb_stmt_error** (resource stmt)

For the statement specified by *stmt*, **maxdb_stmt_error()** returns a containing the error message for the most recently invoked statement function that can succeed or fail.

Return values

A string that describes the error. An empty string if no error occurred.

See also

maxdb_stmt_errno(), **maxdb_stmt_sqlstate()**

Example

Ejemplo 1. Procedural style

```
<?php
/* Open a connection */
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

maxdb_query($link, "CREATE TABLE temp.mycity LIKE hotel.city");
maxdb_query($link, "INSERT INTO temp.mycity SELECT * FROM hotel.city");


$query = "SELECT name, zip FROM temp.mycity ORDER BY name";
if ($stmt = maxdb_prepare($link, $query)) {

   /* drop table */
   maxdb_query($link, "DROP TABLE temp.mycity");

   /* execute query */
   maxdb_stmt_execute($stmt);

   printf("Error: %s.\n", maxdb_stmt_error($stmt));

   /* close statement */
   maxdb_stmt_close($stmt);
}

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
Warning: maxdb_stmt_execute(): -4004 POS(23) Unknown table name:MYCITY [42000] <...>
Error: POS(23) Unknown table name:MYCITY.
```

maxdb_stmt_execute

(no version information, might be only in CVS)

maxdb_stmt_execute -- Executes a prepared Query

Description

Procedural style:

bool **maxdb_stmt_execute** (resource stmt)

The **maxdb_stmt_execute()** function executes a query that has been previously prepared using the **maxdb_prepare()** function represented by the *stmt* resource. When executed any parameter markers which exist will automatically be replaced with the appropiate data.

If the statement is UPDATE, DELETE, or INSERT, the total number of affected rows can be determined by using the **maxdb_stmt_affected_rows()** function. Likewise, if the query yields a result set the **maxdb_fetch()** function is used.

> **Nota:** When using **maxdb_stmt_execute()**, the **maxdb_fetch()** function must be used to fetch the data prior to preforming any additional queries.

Return values

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

See also

maxdb_prepare() **maxdb_stmt_bind_param()**.

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

maxdb_query($link, "CREATE TABLE temp.mycity LIKE hotel.city");

/* Prepare an insert statement */
$query = "INSERT INTO temp.mycity (zip, name, state) VALUES (?,?,?)";
$stmt = maxdb_prepare($link, $query);

maxdb_stmt_bind_param($stmt, "sss", $val1, $val2, $val3);

$val1 = '11111';
$val2 = 'Georgetown';
$val3 = 'NY';

/* Execute the statement */
maxdb_stmt_execute($stmt);

$val1 = '22222';
$val2 = 'Hubbatown';
$val3 = 'CA';

/* Execute the statement */
maxdb_stmt_execute($stmt);

/* close statement */
maxdb_stmt_close($stmt);

/* retrieve all rows from myCity */
$query = "SELECT zip, name, state FROM temp.mycity";
if ($result = maxdb_query($link, $query)) {
   while ($row = maxdb_fetch_row($result)) {
       printf("%s (%s,%s)\n", $row[0], $row[1], $row[2]);
   }
   /* free result set */
   maxdb_free_result($result);
}

/* remove table */
maxdb_query($link, "DROP TABLE temp.mycity");

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
11111 (Georgetown,NY)
22222 (Hubbatown,CA)
```

maxdb_stmt_fetch

(no version information, might be only in CVS)

maxdb_stmt_fetch -- Fetch results from a prepared statement into the bound variables

Description

Procedural style:

mixed **maxdb_stmt_fetch** (resource stmt)

maxdb_stmt_fetch() returns row data using the variables bound by **maxdb_stmt_bind_result()**.

> **Nota:** Note that all columns must be bound by the application before calling **maxdb_stmt_fetch()**.

Return values

Tabla 1. Return values

Value	Description
TRUE	Success. Data has been fetched
FALSE	Error occured
NULL	No more rows/data exists

See also

maxdb_prepare(), **maxdb_stmt_errno()**, **maxdb_stmt_error()**, **maxdb_stmt_bind_result()**

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

$query = "SELECT zip, name FROM hotel.city ORDER by name";

if ($stmt = maxdb_prepare($link, $query)) {

   /* execute statement */
   maxdb_stmt_execute($stmt);

   /* bind result variables */
   maxdb_stmt_bind_result($stmt, $name, $code);

   /* fetch values */
   while (maxdb_stmt_fetch($stmt)) {
       printf ("%s (%s)\n", $name, $code);
   }

   /* close statement */
   maxdb_stmt_close($stmt);
}

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
12203 (Albany)
60601 (Chicago)
60615 (Chicago)
45211 (Cincinnati)
33575 (Clearwater)
75243 (Dallas)
32018 (Daytona Beach)
33441 (Deerfield Beach)
48226 (Detroit)
90029 (Hollywood)
92714 (Irvine)
90804 (Long Beach)
11788 (Long Island)
90018 (Los Angeles)
70112 (New Orleans)
10019 (New York)
10580 (New York)
92262 (Palm Springs)
97213 (Portland)
60018 (Rosemont)
95054 (Santa Clara)
20903 (Silver Spring)
20005 (Washington)
20019 (Washington)
20037 (Washington)
```

maxdb_stmt_free_result

(no version information, might be only in CVS)

maxdb_stmt_free_result -- Frees stored result memory for the given statement handle

Description

Procedural style:

void **maxdb_stmt_free_result** (resource stmt)

The **maxdb_stmt_free_result()** function frees the result memory associated with the statement represented by the *stmt* parameter, which was allocated by **maxdb_stmt_store_result()**.

Valores retornados

This function doesn't return any value.

Ver también

maxdb_stmt_store_result()

maxdb_stmt_init

(no version information, might be only in CVS)

maxdb_stmt_init -- Initializes a statement and returns an resource for use with maxdb_stmt_prepare

Description

Procedural style :

resource **maxdb_stmt_init** (resource link)

Allocates and initializes a statement resource suitable for **maxdb_stmt_prepare()**.

> **Nota:** Any subsequent calls to any maxdb_stmt function will fail until **maxdb_stmt_prepare()** was called.

Valores retornados

Returns an resource.

Ver también

maxdb_stmt_prepare()

maxdb_stmt_num_rows

(no version information, might be only in CVS)

maxdb_stmt_num_rows -- Return the number of rows in statements result set

Description

Procedural style :

mixed **maxdb_stmt_num_rows** (resource stmt)

Returns the number of rows in the result set.

Return values

An integer representing the number of rows in result set.

See also

maxdb_stmt_affected_rows(), **maxdb_prepare()**, **maxdb_stmt_store_result()**

Example

Ejemplo 1. Procedural style

```
<?php
/* Open a connection */
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

$query = "SELECT zip, name FROM hotel.city ORDER BY name";
if ($stmt = maxdb_prepare($link, $query)) {

   /* execute query */
   maxdb_stmt_execute($stmt);

   /* store result */
   maxdb_stmt_store_result($stmt);

   printf("Number of rows: %d.\n", maxdb_stmt_num_rows($stmt));

   /* close statement */
   maxdb_stmt_close($stmt);
}

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
Number of rows: -1.
```

maxdb_stmt_param_count

(no version information, might be only in CVS)

maxdb_stmt_param_count -- Returns the number of parameter for the given statement

Description

Procedural style:

int **maxdb_stmt_param_count** (resource stmt)

maxdb_stmt_param_count() returns the number of parameter markers present in the prepared statement.

Valores retornados

returns an integer representing the number of parameters.

Ver también

maxdb_prepare()

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

if ($stmt = maxdb_prepare($link, "SELECT name FROM hotel.city WHERE name=? OR state=?")

   $marker = maxdb_stmt_param_count($stmt);
   printf("Statement has %d markers.\n", $marker);

   /* close statement */
   maxdb_stmt_close($stmt);
}

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
Statement has 2 markers.
```

maxdb_stmt_prepare

(no version information, might be only in CVS)

maxdb_stmt_prepare -- Prepare a SQL statement for execution

Description

Procedure style:

bool **maxdb_stmt_prepare** (resource stmt, string query)

maxdb_stmt_prepare() prepares the SQL query pointed to by the null-terminated string query. The statement resource has to be allocated by **maxdb_stmt_init()**. The query must consist of a single SQL statement.

> **Nota:** You should not add a terminating semicolon or \g to the statement.

The parameter *query* can include one or more parameter markers in the SQL statement by embedding question mark (*?*) characters at the appropriate positions.

> **Nota:** The markers are legal only in certain places in SQL statements. For example, they are allowed in the VALUES() list of an INSERT statement (to specify column values for a row), or in a comparison with a column in a WHERE clause to specify a comparison value.
>
> However, they are not allowed for identifiers (such as table or column names), in the select list that names the columns to be returned by a SELECT statement), or to specify

both operands of a binary operator such as the = equal sign. The latter restriction is necessary because it would be impossible to determine the parameter type. In general, parameters are legal only in Data Manipulation Languange (DML) statements, and not in Data Defination Language (DDL) statements.

The parameter markers must be bound to application variables using **maxdb_stmt_bind_param()** and/or **maxdb_stmt_bind_result()** before executing the statement or fetching rows.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ver también

maxdb_stmt_init(), **maxdb_stmt_execute()**, **maxdb_stmt_fetch()**, **maxdb_stmt_bind_param()**, **maxdb_stmt_bind_result()**, **maxdb_stmt_close()**

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

$city = "Portland";

/* create a prepared statement */
$stmt = maxdb_stmt_init($link);
if (maxdb_stmt_prepare($stmt, "SELECT state FROM hotel.city WHERE name=?")) {

   /* bind parameters for markers */
   maxdb_stmt_bind_param($stmt, "s", $city);

   /* execute query */
   maxdb_stmt_execute($stmt);

   /* bind result variables */
   maxdb_stmt_bind_result($stmt, $district);

   /* fetch value */
   maxdb_stmt_fetch($stmt);

   printf("%s is in district %s\n", $city, $district);

   /* close statement */
   maxdb_stmt_close($stmt);
}

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
Portland is in district OR
```

maxdb_stmt_reset

(no version information, might be only in CVS)

maxdb_stmt_reset -- Resets a prepared statement

Description

Procedural style:

bool **maxdb_stmt_reset** (resource stmt)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

maxdb_stmt_result_metadata

(no version information, might be only in CVS)

maxdb_stmt_result_metadata -- Returns result set metadata from a prepared statement

Description

Procedural style:

mixed **maxdb_stmt_result_metadata** (resource stmt)

If a statement passed to **maxdb_prepare()** is one that produces a result set, **maxdb_stmt_result_metadata()** returns the result resource that can be used to process the meta information such as total number of fields and individual field information.

> **Nota:** This result set pointer can be passed as an argument to any of the field-based functions that process result set metadata, such as:
>
> - **maxdb_num_fields()**
> - **maxdb_fetch_field()**
> - **maxdb_fetch_field_direct()**
> - **maxdb_fetch_fields()**
> - **maxdb_field_count()**
> - **maxdb_field_seek()**
> - **maxdb_field_tell()**

- maxdb_free_result()

The result set structure should be freed when you are done with it, which you can do by passing it to maxdb_free_result()

> **Nota:** The result set returned by **maxdb_stmt_result_metadata()** contains only metadata. It does not contain any row results. The rows are obtained by using the statement handle with maxdb_fetch().

Valores retornados

maxdb_stmt_result_metadata() returns a result resource or **FALSE** if an error occured.

See also:

maxdb_prepare(), maxdb_free_result()

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

maxdb_query($link, "CREATE TABLE temp.friends (id int, name varchar(20))");

maxdb_query($link, "INSERT INTO temp.friends VALUES (1,'Hartmut')");
maxdb_query($link, "INSERT INTO temp.friends VALUES (2, 'Ulf')");

$stmt = maxdb_prepare($link, "SELECT id, name FROM temp.friends");
maxdb_stmt_execute($stmt);

/* get resultset for metadata */
$result = maxdb_stmt_result_metadata($stmt);

/* retrieve field information from metadata result set */
$field = maxdb_fetch_field($result);

printf("Fieldname: %s\n", $field->name);

/* close resultset */
maxdb_free_result($result);

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
Fieldname: ID
```

maxdb_stmt_send_long_data

(no version information, might be only in CVS)

maxdb_stmt_send_long_data -- Send data in blocks

Description

Procedural style:

bool **maxdb_stmt_send_long_data** (resource stmt, int param_nr, string data)

Allows to send parameter data to the server in pieces (or chunks). This function can be called multiple times to send the parts of a character or binary data value for a column, which must be one of the TEXT or BLOB datatypes.

param_nr indicates which parameter to associate the data with. Parameters are numbered beginning with 0. *data* is a string containing data to be sent.

> **Nota:** For efficiency reasons, this function should be used after calling **maxdb_execute ()**. In this case, the data is not stored on the client side. The end of the sequence must end with a call to **maxdb_stmt_close_long_data()**.

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Ver también

maxdb_prepare(), **maxdb_stmt_bind_param()**

maxdb_stmt_sqlstate

(no version information, might be only in CVS)

maxdb_stmt_sqlstate -- Returns SQLSTATE error from previous statement operation

Description

string **maxdb_stmt_sqlstate** (resource stmt)

Returns a string containing the SQLSTATE error code for the most recently invoked prepared statement function that can succeed or fail. The error code consists of five characters. *'00000'* means no error. The values are specified by ANSI SQL and ODBC.

> **Nota:** Note that not all MaxDB errors are yet mapped to SQLSTATE's. The value *HY000* (general error) is used for unmapped errors.

Return values

Returns a string containing the SQLSTATE error code for the last error. The error code consists of five characters. *'00000'* means no error.

See also

maxdb_stmt_errno(), **maxdb_stmt_error()**

Example

Ejemplo 1. Procedural style

```
<?php
/* Open a connection */
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

maxdb_query($link, "CREATE TABLE temp.mycity LIKE hotel.city");
maxdb_query($link, "INSERT INTO temp.mycity SELECT * FROM hotel.city");

$query = "SELECT name, zip FROM temp.mycity ORDER BY name";
if ($stmt = maxdb_prepare($link, $query)) {

   /* drop table */
   maxdb_query($link, "DROP TABLE temp.mycity");

   /* execute query */
   maxdb_stmt_execute($stmt);

   printf("Error: %s.\n", maxdb_stmt_sqlstate($stmt));

   /* close statement */
   maxdb_stmt_close($stmt);
}

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
Warning: maxdb_stmt_execute(): -4004 POS(23) Unknown table name:MYCITY [42000] <...>
Error: 42000.
```

maxdb_stmt_store_result

(no version information, might be only in CVS)

maxdb_stmt_store_result -- Transfers a result set from a prepared statement

Description

Procedural style:

bool **maxdb_stmt_store_result** (resource stmt)

maxdb_stmt_store_result() has no functionally effect and should not be used for retrieving data from MaxDB server.

Return values

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

See also

maxdb_prepare(), **maxdb_stmt_result_metadata()**, **maxdb_fetch()**

Example

Ejemplo 1. Procedural style

```
<?php
/* Open a connection */
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

$query = "SELECT name, zip FROM hotel.city ORDER BY name";
if ($stmt = maxdb_prepare($link, $query)) {

   /* execute query */
   maxdb_stmt_execute($stmt);

   /* store result */
   maxdb_stmt_store_result($stmt);

   printf("Number of rows: %d.\n", maxdb_stmt_num_rows($stmt));

   /* free result */
   maxdb_stmt_free_result($stmt);

   /* close statement */
   maxdb_stmt_close($stmt);
}

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
Number of rows: -1.
```

maxdb_store_result

(no version information, might be only in CVS)

maxdb_store_result -- Transfers a result set from the last query

Description

Procedural style:

resource **maxdb_store_result** (resource link)

This function has no functionally effect.

Return values

Returns a result resource or **`FALSE`** if an error occurred.

See also

maxdb_real_query(), **maxdb_use_result()**.

Example

See **maxdb_multi_query()**.

maxdb_thread_id

(no version information, might be only in CVS)

maxdb_thread_id -- Returns the thread ID for the current connection

Description

Procedural style:

int **maxdb_thread_id** (resource link)

The **maxdb_thread_id()** function returns the thread ID for the current connection which can then be killed using the **maxdb_kill()** function. If the connection is lost and you reconnect with **maxdb_ping()**, the thread ID will be other. Therefore you should get the thread ID only when you need it.

> **Nota:** The thread ID is assigned on a connection-by-connection basis. Hence, if the connection is broken and then re-established a new thread ID will be assigned.

Return values

maxdb_thread_id() returns the Thread ID for the current connection.

See also

maxdb_kill()

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

/* determine our thread id */
$thread_id = maxdb_thread_id($link);

/* Kill connection */
maxdb_kill($link, $thread_id);

/* This should produce an error */
if (!maxdb_query($link, "CREATE TABLE mycity LIKE hotel.city")) {
   printf("Error: %s\n", maxdb_error($link));
   exit;
}

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
Warning: maxdb_query(): -10821 Session not connected <...>
Error: Session not connected
```

maxdb_thread_safe

(no version information, might be only in CVS)

maxdb_thread_safe -- Returns whether thread safety is given or not

Description

Procedural style:

bool **maxdb_thread_safe** (void)

maxdb_thread_safe() indicates whether the client library is compiled as thread-safe.

Return values

TRUE if the client library is thread-safe, otherwise **FALSE**.

maxdb_use_result

(no version information, might be only in CVS)

maxdb_use_result -- Initiate a result set retrieval

Description

Procedural style:

mixed **maxdb_use_result** (resource link)

maxdb_use_result() has no effect.

Return values

Returns always **FALSE**.

See also

maxdb_real_query(), maxdb_store_result().

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

$query  = "SELECT * FROM DUAL;";
$query .= "SELECT Name FROM City ORDER BY ID LIMIT 20, 5";

/* execute multi query */
if (maxdb_multi_query($link, $query)) {
   do {
       /* store first result set */
       if ($result = maxdb_use_result($link)) {
           while ($row = maxdb_fetch_row($result)) {
               printf("%s\n", $row[0]);
           }
           maxdb_free_result($result);
       }
       /* print divider */
       if (maxdb_more_results($link)) {
           printf("-----------------\n");
       }
   } while (maxdb_next_result($link));
}

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
Warning: maxdb_multi_query(): -3008 POS(19) Invalid keyword or missing delimiter <...>
```

maxdb_warning_count

(no version information, might be only in CVS)

maxdb_warning_count -- Returns the number of warnings from the last query for the given link

Description

Procedural style:

int **maxdb_warning_count** (resource link)

maxdb_warning_count() returns the number of warnings from the last query in the connection represented by the *link* parameter.

Return values

Number of warnings or zero if there are no warnings.

See also

maxdb_errno(), **maxdb_error()**, **maxdb_sqlstate()**

Example

Ejemplo 1. Procedural style

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED");

/* check connection */
if (maxdb_connect_errno()) {
   printf("Connect failed: %s\n", maxdb_connect_error());
   exit();
}

maxdb_query($link, "CREATE TABLE temp.mycity LIKE hotel.city");

/* a remarkable long city name in Wales */
$query = "INSERT INTO temp.mycity (zip, name) VALUES('11111',
       'Llanfairpwllgwyngyllgogerychwyrndrobwllllantysiliogogogoch')";

maxdb_query($link, $query);

printf ("Number of warning: %d\n", maxdb_warning_count($link));

/* close connection */
maxdb_close($link);
?>
```

The above examples would produce the following output:

```
Warning: maxdb_query(): -8004 POS(62) Constant must be compatible with column type and
Number of warning: 0
```

LXV. Multibyte String Functions

Introducción

While there are many languages in which every necessary character can be represented by a one-to-one mapping to a 8-bit value, there are also several languages which require so many characters for written communication that cannot be contained within the range a mere byte can code. Multibyte character encoding schemes were developed to express that many (more than 256) characters in the

regular bytewise coding system.

When you manipulate (trim, split, splice, etc.) strings encoded in a multibyte encoding, you need to use special functions since two or more consecutive bytes may represent a single character in such encoding schemes. Otherwise, if you apply a non-multibyte-aware string function to the string, it probably fails to detect the beginning or ending of the multibyte character and ends up with a corrupted garbage string that most likely loses its original meaning.

mbstring provides these multibyte specific string functions that help you deal with multibyte encodings in PHP, which is basically supposed to be used with single byte encodings. In addition to that, *mbstring* handles character encoding conversion between the possible encoding pairs.

mbstring is also designed to handle Unicode-based encodings such as UTF-8 and UCS-2 and many single-byte encodings for convenience (listed below), whereas *mbstring* was originally developed for use in Japanese web pages.

PHP Character Encoding Requirements

Encodings of the following types are safely used with PHP.

- A singlebyte encoding,
    - which has ASCII-compatible (ISO646 compatible) mappings for the characters in range of *00h* to *7fh*.
- A multibyte encoding,
    - which has ASCII-compatible mappings for the characters in range of *00h* to *7fh*.
    - which don't use ISO2022 escape sequences.
    - which don't use a value from *00h* to *7fh* in any of the compounded bytes that represents a single character.

These are examples of character encodings that are unlikely to work with PHP.

```
JIS, SJIS, ISO-2022-JP, BIG-5
```

Although PHP scripts written in any of those encodings might not work, especially in the case where encoded strings appear as identifiers or literals in the script, you can almost avoid using these encodings by setting up the *mbstring*'s transparent encoding filter function for incoming HTTP queries.

> **Nota:** It's highly discouraged to use SJIS, BIG5, CP936, CP949 and GB18030 for the internal encoding unless you are familiar with the parser, the scanner and the character encoding.
>
> **Nota:** If you have some database connected with PHP, it is recommended that you use the same character encoding for both database and the *internal encoding* for ease of use and better performance.
>
> If you are using PostgreSQL, the character encoding used in the database and the one used in the PHP may differ as it supports automatic character set conversion between

the backend and the frontend.

Instalación

mbstring is a non-default extension. This means it is not enabled by default. You must explicitly enable the module with the *configure* option. See the Install section for details.

The following configure options are related to the *mbstring* module.

- *--enable-mbstring=LANG*: Enable *mbstring* functions. This option is required to use *mbstring* functions.

  As of PHP 4.3.0, *mbstring* extension provides enhanced support for Simplified Chinese, Traditional Chinese, Korean, and Russian in addition to Japanese. To enable that feature, you will have to supply either one of the following options to the *LANG* parameter; *--enable-mbstring=cn* for Simplified Chinese support, *--enable-mbstring=tw* for Traditional Chinese support, *--enable-mbstring=kr* for Korean support, *--enable-mbstring=ru* for Russian support, and *--enable-mbstring=ja* for Japanese support.

  Also *--enable-mbstring=all* is convenient for you to enable all the supported languages listed above.

  > **Nota:** Japanese language support is also enabled by *--enable-mbstring* without any options for the sake of backwards compatibility.

- *--enable-mbstr-enc-trans* : Enable HTTP input character encoding conversion using *mbstring* conversion engine. If this feature is enabled, HTTP input character encoding may be converted to *mbstring.internal_encoding* automatically.

  > **Nota:** As of PHP 4.3.0, the option *--enable-mbstr-enc-trans* was eliminated and replaced with the runtime setting *mbstring.encoding_translation*. HTTP input character encoding conversion is enabled when this is set to *On* (the default is *Off*).

- *--enable-mbregex*: Enable regular expression functions with multibyte character support.

Configuración en tiempo de ejecución

El comportamiento de estas funciones está afectado por los valores definidos en `php.ini`.

Tabla 1. mbstring configuration options

Name	Default	Changeable
mbstring.language	"neutral"	PHP_INI_SYSTEM | PHP_INI_PERDIR
mbstring.detect_order	NULL	PHP_INI_ALL
mbstring.http_input	"pass"	PHP_INI_ALL
mbstring.http_output	"pass"	PHP_INI_ALL

Name	Default	Changeable
mbstring.internal_encoding	NULL	PHP_INI_ALL
mbstring.script_encoding	NULL	PHP_INI_ALL
mbstring.substitute_character	NULL	PHP_INI_ALL
mbstring.func_overload	"0"	PHP_INI_SYSTEM | PHP_INI_PERDIR
mbstring.encoding_translation	"0"	PHP_INI_SYSTEM | PHP_INI_PERDIR

For the definition of the PHP_INI_* constants, please refer to **ini_set()**.

A continuación se presenta una corta explicación de las directivas de configuración.

mbstring.language **string**

The default national language setting (NLS) used in mbstring. Note that this option automagically defines *mbstring.internal_encoding* and *mbstring.internal_encoding* should be placed after *mbstring.language* in `php.ini`

mbstring.encoding_translation **boolean**

Enables the transparent character encoding filter for the incoming HTTP queries, which performs detection and conversion of the input encoding to the internal character encoding.

mbstring.internal_encoding **string**

Defines the default internal character encoding.

mbstring.http_input **string**

Defines the default HTTP input character encoding.

mbstring.http_output **string**

Defines the default HTTP output character encoding.

mbstring.detect_order **string**

Defines default character code detection order. See also **mb_detect_order()**.

mbstring.substitute_character **string**

Defines character to substitute for invalid character encoding.

mbstring.func_overload **string**

Overloads a set of single byte functions by the mbstring counterparts. See Function overloading for more information.

According to the HTML 4.01 specification, Web browsers are allowed to encode a form being submitted with a character encoding different from the one used for the page. See **mb_http_input()** to detect character encoding used by browsers.

Although popular browsers are capable of giving a reasonably accurate guess to the character encoding of a given HTML document, it would be better to set the *charset* parameter in the *Content-Type* HTTP header to the appropriate value by **header()** or default_charset ini setting.

Ejemplo 1. `php.ini` setting examples

```
; Set default language
mbstring.language        = Neutral; Set default language to Neutral(UTF-8) (default)
mbstring.language        = English; Set default language to English
mbstring.language        = Japanese; Set default language to Japanese

;; Set default internal encoding
;; Note: Make sure to use character encoding works with PHP
mbstring.internal_encoding    = UTF-8  ; Set internal encoding to UTF-8

;; HTTP input encoding translation is enabled.
mbstring.encoding_translation = On

;; Set default HTTP input character encoding
;; Note: Script cannot change http_input setting.
mbstring.http_input           = pass    ; No conversion.
mbstring.http_input           = auto    ; Set HTTP input to auto
                                ; "auto" is expanded to "ASCII,JIS,UTF-8,EUC-JP,SJIS"
mbstring.http_input           = SJIS    ; Set HTTP2 input to  SJIS
mbstring.http_input           = UTF-8,SJIS,EUC-JP ; Specify order

;; Set default HTTP output character encoding
mbstring.http_output          = pass    ; No conversion
mbstring.http_output          = UTF-8   ; Set HTTP output encoding to UTF-8

;; Set default character encoding detection order
mbstring.detect_order         = auto    ; Set detect order to auto
mbstring.detect_order         = ASCII,JIS,UTF-8,SJIS,EUC-JP ; Specify order

;; Set default substitute character
mbstring.substitute_character = 12307   ; Specify Unicode value
mbstring.substitute_character = none    ; Do not print character
mbstring.substitute_character = long    ; Long Example: U+3000,JIS+7E7E
```

Ejemplo 2. `php.ini` setting for *EUC-JP* users

```
;; Disable Output Buffering
output_buffering      = Off

;; Set HTTP header charset
default_charset       = EUC-JP

;; Set default language to Japanese
mbstring.language = Japanese

;; HTTP input encoding translation is enabled.
mbstring.encoding_translation = On

;; Set HTTP input encoding conversion to auto
mbstring.http_input   = auto

;; Convert HTTP output to EUC-JP
mbstring.http_output  = EUC-JP

;; Set internal encoding to EUC-JP
mbstring.internal_encoding = EUC-JP

;; Do not print invalid characters
mbstring.substitute_character = none
```

Ejemplo 3. `php.ini` setting for *SJIS* users

```
;; Enable Output Buffering
output_buffering     = On

;; Set mb_output_handler to enable output conversion
output_handler       = mb_output_handler

;; Set HTTP header charset
default_charset      = Shift_JIS

;; Set default language to Japanese
mbstring.language = Japanese

;; Set http input encoding conversion to auto
mbstring.http_input  = auto

;; Convert to SJIS
mbstring.http_output = SJIS

;; Set internal encoding to EUC-JP
mbstring.internal_encoding = EUC-JP

;; Do not print invalid characters
mbstring.substitute_character = none
```

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

MB_OVERLOAD_MAIL (integer)

MB_OVERLOAD_STRING (integer)

MB_OVERLOAD_REGEX (integer)

HTTP Input and Output

HTTP input/output character encoding conversion may convert binary data also. Users are supposed to control character encoding conversion if binary data is used for HTTP input/output.

> **Nota:** In PHP 4.3.2 or earlier versions, there was a limitation in this functionality that *mbstring* does not perform character encoding conversion in POST data if the *enctype* attribute in the *form* element is set to *multipart/form-data*. So you have to convert the incoming data by yourself in this case if necessary.
>
> Beginning with PHP 4.3.3, if *enctype* for HTML form is set to *multipart/form-data* and *mbstring.encoding_translation* is set to On in `php.ini` the POST'ed variables and the names of uploaded files will be converted to the internal character encoding as well.

However, the conversion isn't applied to the query keys.

- HTTP Input

  There is no way to control HTTP input character conversion from PHP script. To disable HTTP input character conversion, it has to be done in `php.ini`.

  Ejemplo 4. Disable HTTP input conversion in `php.ini`

```
;; Disable HTTP Input conversion
mbstring.http_input = pass
;; Disable HTTP Input conversion (PHP 4.3.0 or higher)
mbstring.encoding_translation = Off
```

  When using PHP as an Apache module, it is possible to override those settings in each Virtual Host directive in `httpd.conf` or per directory with `.htaccess`. Refer to the Configuration section and Apache Manual for details.

- HTTP Output

  There are several ways to enable output character encoding conversion. One is using `php.ini`, another is using **ob_start()** with **mb_output_handler()** as *ob_start* callback function.

  > **Nota:** PHP3-i18n users should note that *mbstring*'s output conversion differs from PHP3-i18n. Character encoding is converted using output buffer.

Ejemplo 5. `php.ini` setting example

```
;; Enable output character encoding conversion for all PHP pages

;; Enable Output Buffering
output_buffering    = On

;; Set mb_output_handler to enable output conversion
output_handler      = mb_output_handler
```

Ejemplo 6. Script example

```
<?php

// Enable output character encoding conversion only for this page

// Set HTTP output character encoding to SJIS
mb_http_output('SJIS');

// Start buffering and specify "mb_output_handler" as
// callback function
ob_start('mb_output_handler');

?>
```

Supported Character Encodings

Currently the following character encodings are supported by the *mbstring* module. Any of those Character encodings can be specified in the *encoding* parameter of *mbstring* functions.

The following character encoding is supported in this PHP extension:

- UCS-4
- UCS-4BE

- UCS-4LE
- UCS-2
- UCS-2BE
- UCS-2LE
- UTF-32
- UTF-32BE
- UTF-32LE
- UTF-16
- UTF-16BE
- UTF-16LE
- UTF-7
- UTF7-IMAP
- UTF-8
- ASCII
- EUC-JP
- SJIS
- eucJP-win
- SJIS-win
- ISO-2022-JP
- JIS
- ISO-8859-1
- ISO-8859-2
- ISO-8859-3
- ISO-8859-4
- ISO-8859-5
- ISO-8859-6
- ISO-8859-7

- ISO-8859-8
- ISO-8859-9
- ISO-8859-10
- ISO-8859-13
- ISO-8859-14
- ISO-8859-15
- byte2be
- byte2le
- byte4be
- byte4le
- BASE64
- HTML-ENTITIES
- 7bit
- 8bit
- EUC-CN
- CP936
- HZ
- EUC-TW
- CP950
- BIG-5
- EUC-KR
- UHC (CP949)
- ISO-2022-KR
- Windows-1251 (CP1251)
- Windows-1252 (CP1252)
- CP866 (IBM866)
- KOI8-R

`php.ini` entry, which accepts encoding name, accepts "*auto*" and "*pass*" also. *mbstring* functions, which accepts encoding name, and accepts "*auto*".

If "*pass*" is set, no character encoding conversion is performed.

If "*auto*" is set, it is expanded to the list of encodings defined per the NLS. For instance, if the NLS is set to *Japanese*, the value is assumed to be "*ASCII,JIS,UTF-8,EUC-JP,SJIS*".

See also **mb_detect_order()**

Function Overloading Feature

You might often find it difficult to get an existing PHP application work in a given multibyte environment. That's mostly because lots of PHP applications out there are written with the standard string functions such as **substr()**, which are known to not properly handle multibyte-encoded strings.

mbstring supports 'function overloading' feature which enables you to add multibyte awareness to such an application without code modification by overloading multibyte counterparts on the standard string functions. For example, **mb_substr()** is called instead of **substr()** if function overloading is enabled. This feature makes it easy to port applications that only support single-byte encodings to a multibyte environment in many cases.

To use the function overloading, set *mbstring.func_overload* in `php.ini` to a positive value that represents a combination of bitmasks specifying the categories of functions to be overloaded. It should be set to 1 to overload the **mail()** function. 2 for string functions, 4 for regular expression functions. For example, if is set for 7, mail, strings and regular expression functions should be overloaded. The list of overloaded functions are shown below.

Tabla 2. Functions to be overloaded

value of mbstring.func_overload	original function	overloaded function
1	**mail()**	**mb_send_mail()**
2	**strlen()**	**mb_strlen()**
2	**strpos()**	**mb_strpos()**
2	**strrpos()**	**mb_strrpos()**
2	**substr()**	**mb_substr()**
2	**strtolower()**	**mb_strtolower()**
2	**strtoupper()**	**mb_strtoupper()**
2	**substr_count()**	**mb_substr_count()**
4	**ereg()**	**mb_ereg()**
4	**eregi()**	**mb_eregi()**
4	**ereg_replace()**	**mb_ereg_replace()**
4	**eregi_replace()**	**mb_eregi_replace()**
4	**split()**	**mb_split()**

Nota: It is not recommended to use the function overloading option in the per-directory

context, because it's not confirmed yet to be stable enough in a production environment and may lead to undefined behaviour.

Basics of Japanese multi-byte encodings

It is often said quite hard to figure out how Japanese texts are handled in the computer. This is not only because Japanese characters can only be represented by multibyte encodings, but because different encoding standards are adopted for different purposes / platforms. Moreover, not a few character set standards are used there, which are slightly different from one another. Those facts have often led developers to inevitable mess-up.

To create a working web application that would be put in the Japanese environment, it is important to use the proper character encoding and character set for the task in hand.

- Storage for a character can be up to six bytes
- Most of multibyte characters often appear twice as wide as a single-byte character on display. Those characters are called "zen-kaku" in Japanese which means "full width", and the other (narrower) characters are called "han-kaku" - means half width. However the graphical properties of the characters depend on the glyphs of the type faces used to display them or print them out.
- Some character encodings use shift(escape) sequences defined in ISO2022 to switch the code map of the specific code area (*00h* to *7fh*).
- ISO-2022-JP should be used in SMTP/NNTP, and headers and entities should be reencoded as per RFC requirements. Although those are not requisites, it's still a good idea because several popular user agents cannot recognize any other encoding methods.
- Webpages created for mobile phone services such as i-mode, Vodafone live!, or EZweb are supposed to use Shift_JIS.

References

Multibyte character encoding schemes and the related issues are very complicated. There should be too few space to cover in sufficient details. Please refer to the following URLs and other resources for further readings.

- Unicode materials

  http://www.unicode.org/

- Japanese/Korean/Chinese character information

  http://examples.oreilly.com/cjkvinfo/doc/cjk.inf

Summaries of supported encodings

Summaries of supported encodings

Name in the IANA character set registry: ISO-10646-UCS-4

Underlying character set: ISO 10646

Description: The Universal Character Set with 31-bit code space, standardized as UCS-4 by ISO/IEC 10646. It is kept synchronized with the latest version of the Unicode code map.

Additional note: If this name is used in the encoding conversion facility, the converter attempts to identify by the preceding BOM (byte order mark)in which endian the subsequent bytes are represented.

Name in the IANA character set registry: ISO-10646-UCS-4

Underlying character set: UCS-4

Description: See above.

Additional note: In contrast to *UCS-4*, strings are always assumed to be in big endian form.

Name in the IANA character set registry: ISO-10646-UCS-4

Underlying character set: UCS-4

Description: See above.

Additional note: In contrast to *UCS-4*, strings are always assumed to be in little endian form.

Name in the IANA character set registry: ISO-10646-UCS-2

Underlying character set: UCS-2

Description: The Universal Character Set with 16-bit code space, standardized as UCS-2 by ISO/IEC 10646. It is kept synchronized with the latest version of the unicode code map.

Additional note: If this name is used in the encoding conversion facility, the converter attempts to identify by the preceding BOM (byte order mark)in which endian the subsequent bytes are represented.

Name in the IANA character set registry: ISO-10646-UCS-2

Underlying character set: UCS-2

Description: See above.

Additional note: In contrast to *UCS-2*, strings are always assumed to be in big endian form.

Name in the IANA character set registry: ISO-10646-UCS-2

Underlying character set: UCS-2

Description: See above.

Additional note: In contrast to *UCS-2*, strings are always assumed to be in little endian form.

Name in the IANA character set registry: UTF-32

Underlying character set: Unicode

Description: Unicode Transformation Format of 32-bit unit width, whose encoding space refers to the Unicode's codeset standard. This encoding scheme wasn't identical to UCS-4 because the code space of Unicode were limited to a 21-bit value.

Additional note: If this name is used in the encoding conversion facility, the converter attempts to identify by the preceding BOM (byte order mark)in which endian the subsequent bytes are represented.

Name in the IANA character set registry: UTF-32BE

Underlying character set: Unicode

Description: See above

Additional note: In contrast to *UTF-32*, strings are always assumed to be in big endian form.

Name in the IANA character set registry: UTF-32LE

Underlying character set: Unicode

Description: See above

Additional note: In contrast to *UTF-32*, strings are always assumed to be in little endian form.

Name in the IANA character set registry: UTF-16

Underlying character set: Unicode

Description: Unicode Transformation Format of 16-bit unit width. It's worth a note that UTF-16 is no longer the same specification as UCS-2 because the surrogate mechanism has been introduced since Unicode 2.0 and UTF-16 now refers to a 21-bit code space.

Additional note: If this name is used in the encoding conversion facility, the converter attempts to identify by the preceding BOM (byte order mark)in which endian the subsequent bytes are represented.

Name in the IANA character set registry: UTF-16BE

Underlying character set: Unicode

Description: See above.

Additional note: In contrast to *UTF-16*, strings are always assumed to be in big endian form.

Name in the IANA character set registry: UTF-16BE

Underlying character set: Unicode

Description: See above.

Additional note: In contrast to *UTF-16*, strings are always assumed to be in big endian form.

Name in the IANA character set registry: UTF-8

Underlying character set: Unicode / UCS

Description: Unicode Transformation Format of 8-bit unit width.

Additional note: none

Name in the IANA character set registry: UTF-7

Underlying character set: Unicode

Description: A mail-safe transformation format of Unicode, specified in RFC2152.

Additional note: none

Name in the IANA character set registry: (none)

Underlying character set: Unicode

Description: A variant of UTF-7 which is specialized for use in the IMAP protocol.

Additional note: none

Name in the IANA character set registry: US-ASCII (preferred MIME name) / iso-ir-6 / ANSI_X3.4-1986 / ISO_646.irv:1991 / ASCII / ISO646-US / us / IBM367 / CP367 / csASCII

Underlying character set: ASCII / ISO 646

Description: American Standard Code for Information Interchange is a commonly-used 7-bit encoding. Also standardized as an international standard, ISO 646.

Additional note: (none)

Name in the IANA character set registry: EUC-JP (preferred MIME name) / Extended_UNIX_Code_Packed_Format_for_Japanese / csEUCPkdFmtJapanese

Underlying character set: Compound of US-ASCII / JIS X0201:1997 (hankaku kana part) / JIS X0208:1990 / JIS X0212:1990

Description: As you see the name is derived from an abbreviation of Extended UNIX Code Packed Format for Japanese, this encoding is mostly used on UNIX or alike platforms. The original encoding scheme, Extended UNIX Code, is designed on the basis of ISO 2022.

Additional note: The character set referred to by EUC-JP is different to IBM932 / CP932, which are used by OS/2Ã‚Â® and MicrosoftÃ‚Â® WindowsÃ‚Â®. For information interchange with those platforms, use EUCJP-WIN instead.

Name in the IANA character set registry: Shift_JIS (preferred MIME name) / MS_Kanji / csShift_JIS

Underlying character set: Compound of JIS X0201:1997 / JIS X0208:1997

Description: Shift_JIS was developed in early 80's, at the time personal Japanese word processors were brought into the market, in order to maintain compatiblities with the legacy encoding scheme JIS X 0201:1976. According to the IANA definition the codeset of Shift_JIS is slightly different to IBM932 / CP932. However, the names "SJIS" / "Shift_JIS" are often wrongly used to refer to these codesets.

Additional note: For the CP932 codemap, use SJIS-WIN instead.

Name in the IANA character set registry: (none)

Underlying character set: Compound of JIS X0201:1997 / JIS X0208:1997 / IBM extensions / NEC extensions

Description: While this "encoding" uses the same encoding scheme as EUC-JP, the underlying character set is different. That is, some code points map to different characters than EUC-JP.

Additional note: none

Name in the IANA character set registry: Windows-31J / csWindows31J

Underlying character set: Compound of JIS X0201:1997 / JIS X0208:1997 / IBM extensions / NEC extensions

Description: While this "encoding" uses the same encoding scheme as Shift_JIS, the underlying character set is different. That means some code points map to different characters than Shift_JIS.

Additional note: (none)

Name in the IANA character set registry: ISO-2022-JP (preferred MIME name) / csISO2022JP

Underlying character set: US-ASCII / JIS X0201:1976 / JIS X0208:1978 / JIS X0208:1983

Description: RFC1468

Additional note: (none)

Name in the IANA character set registry: JIS

Underlying character set:

Description:

Additional note:

Name in the IANA character set registry: ISO-8859-1

Underlying character set:

Description:

Additional note:

Name in the IANA character set registry: ISO-8859-2

Underlying character set:

Description:

Additional note:

Name in the IANA character set registry: ISO-8859-3

Underlying character set:

Description:

Additional note:

Name in the IANA character set registry: ISO-8859-4

Underlying character set:

Description:

Additional note:

Name in the IANA character set registry: ISO-8859-5

Underlying character set:

Description:

Additional note:

Name in the IANA character set registry: ISO-8859-6

Underlying character set:

Description:

Additional note:

Name in the IANA character set registry: ISO-8859-7

Underlying character set:

Description:

Additional note:

Name in the IANA character set registry: ISO-8859-8

Underlying character set:

Description:

Additional note:

Name in the IANA character set registry: ISO-8859-9

Underlying character set:

Description:

Additional note:

Name in the IANA character set registry: ISO-8859-10

Underlying character set:

Description:

Additional note:

Name in the IANA character set registry: ISO-8859-13

Underlying character set:

Description:

Additional note:

Name in the IANA character set registry: ISO-8859-14

Underlying character set:

Description:

Additional note:

Name in the IANA character set registry: ISO-8859-15

Underlying character set:

Description:

Additional note:

Name in the IANA character set registry: byte2be

Underlying character set:

Description:

Additional note:

Name in the IANA character set registry: byte2le

Underlying character set:

Description:

Additional note:

Name in the IANA character set registry: byte4be

Underlying character set:

Description:

Additional note:

Name in the IANA character set registry: byte4le

Underlying character set:

Description:

Additional note:

Name in the IANA character set registry: BASE64

Underlying character set:

Description:

Additional note:

Name in the IANA character set registry: HTML-ENTITIES

Underlying character set:

Description:

Additional note:

Name in the IANA character set registry: 7bit

Underlying character set:

Description:

Additional note:

Name in the IANA character set registry: 8bit

Underlying character set:

Description:

Additional note:

Name in the IANA character set registry: EUC-CN

Underlying character set:

Description:

Additional note:

Name in the IANA character set registry: CP936

Underlying character set:

Description:

Additional note:

Name in the IANA character set registry: HZ

Underlying character set:

Description:

Additional note:

Name in the IANA character set registry: EUC-TW

Underlying character set:

Description:

Additional note:

Name in the IANA character set registry: CP950

Underlying character set:

Description:

Additional note:

Name in the IANA character set registry: BIG-5

Underlying character set:

Description:

Additional note:

Name in the IANA character set registry: EUC-KR

Underlying character set:

Description:

Additional note:

Name in the IANA character set registry: UHC (CP949)

Underlying character set:

Description:

Additional note:

Name in the IANA character set registry: ISO-2022-KR

Underlying character set:

Description:

Additional note:

Name in the IANA character set registry: Windows-1251 (CP1251)

Underlying character set:

Description:

Additional note:

Name in the IANA character set registry: Windows-1252 (CP1252)

Underlying character set:

Description:

Additional note:

Name in the IANA character set registry: CP866 (IBM866)

Underlying character set:

Description:

Additional note:

Name in the IANA character set registry: KOI8-R

Underlying character set:

Description:

Additional note:

Tabla de contenidos

mb_convert_variables -- Convert character code in variable(s)
mb_decode_mimeheader -- Decode string in MIME header field
mb_decode_numericentity -- Decode HTML numeric string reference to character
mb_detect_encoding -- Detect character encoding
mb_detect_order -- Set/Get character encoding detection order
mb_encode_mimeheader -- Encode string for MIME header
mb_encode_numericentity -- Encode character to HTML numeric string reference
mb_ereg_match -- Regular expression match for multibyte string
mb_ereg_replace -- Replace regular expression with multibyte support
mb_ereg_search_getpos -- Returns start point for next regular expression match
mb_ereg_search_getregs -- Retrieve the result from the last multibyte regular expression match
mb_ereg_search_init -- Setup string and regular expression for multibyte regular expression match
mb_ereg_search_pos -- Return position and length of matched part of multibyte regular expression for predefined multibyte string
mb_ereg_search_regs -- Returns the matched part of multibyte regular expression
mb_ereg_search_setpos -- Set start point of next regular expression match
mb_ereg_search -- Multibyte regular expression match for predefined multibyte string
mb_ereg -- Regular expression match with multibyte support
mb_eregi_replace -- Replace regular expression with multibyte support ignoring case
mb_eregi -- Regular expression match ignoring case with multibyte support
mb_get_info -- Get internal settings of mbstring
mb_http_input -- Detect HTTP input character encoding
mb_http_output -- Set/Get HTTP output character encoding
mb_internal_encoding -- Set/Get internal character encoding
mb_language -- Set/Get current language
mb_list_encodings -- Returns an array of all supported encodings
mb_output_handler -- Callback function converts character encoding in output buffer
mb_parse_str -- Parse GET/POST/COOKIE data and set global variable
mb_preferred_mime_name -- Get MIME charset string
mb_regex_encoding -- Returns current encoding for multibyte regex as string
mb_regex_set_options -- Set/Get the default options for mbregex functions
mb_send_mail -- Send encoded mail
mb_split -- Split multibyte string using regular expression
mb_strcut -- Get part of string
mb_strimwidth -- Get truncated string with specified width
mb_strlen -- Get string length
mb_strpos -- Find position of first occurrence of string in a string
mb_strrpos -- Find position of last occurrence of a string in a string
mb_strtolower -- Make a string lowercase
mb_strtoupper -- Make a string uppercase
mb_strwidth -- Return width of string
mb_substitute_character -- Set/Get substitution character
mb_substr_count -- Count the number of substring occurrences
mb_substr -- Get part of string

mb_convert_case

(PHP 4 >= 4.3.0, PHP 5)

mb_convert_case -- Perform case folding on a string

Description

string **mb_convert_case** (string str, int mode [, string encoding])

mb_convert_case() returns case folded version of *string* converted in the way specified by *mode*.

mode can be one of MB_CASE_UPPER, MB_CASE_LOWER or MB_CASE_TITLE.

encoding specifies the encoding of *str*; if omitted, the internal character encoding value will be used.

The return value is str with the appropriate case folding applied.

By contrast to the standard case folding functions such as **strtolower()** and **strtoupper()**, case folding is performed on the basis of the Unicode character properties. Thus the behaviour of this function is not affected by locale settings and it can convert any characters that have 'alphabetic' property, such as A-umlaut (Ã*f*â€ž).

For more information about the Unicode properties, please see http://www.unicode.org/unicode/reports/tr21/.

Ejemplo 1. mb_convert_case() example

```
<?php
$str = "mary had a Little lamb and she loved it so";
$str = mb_convert_case($str, MB_CASE_UPPER, "UTF-8");
echo $str; // Prints MARY HAD A LITTLE LAMB AND SHE LOVED IT SO
$str = mb_convert_case($str, MB_CASE_TITLE, "UTF-8");
echo $str; // Prints Mary Had A Little Lamb And She Loved It So
?>
```

See also **mb_strtolower()**, **mb_strtoupper()**, **strtolower()**, **strtoupper()**, **ucfirst()**, y **ucwords()**

mb_convert_encoding

(PHP 4 >= 4.0.6, PHP 5)

mb_convert_encoding -- Convert character encoding

Description

string **mb_convert_encoding** (string str, string to_encoding [, mixed from_encoding])

mb_convert_encoding() converts character encoding of string *str* from *from_encoding* to *to_encoding*.

str : String to be converted.

from_encoding is specified by character code name before conversion. it can be array or string - comma separated enumerated list. If it is not specified, the internal encoding will be used.

Ejemplo 1. mb_convert_encoding() example

```php
<?php
/* Convert internal character encoding to SJIS */
$str = mb_convert_encoding($str, "SJIS");

/* Convert EUC-JP to UTF-7 */
$str = mb_convert_encoding($str, "UTF-7", "EUC-JP");

/* Auto detect encoding from JIS, eucjp-win, sjis-win, then convert str to UCS-2LE */
$str = mb_convert_encoding($str, "UCS-2LE", "JIS, eucjp-win, sjis-win");

/* "auto" is expanded to "ASCII,JIS,UTF-8,EUC-JP,SJIS" */
$str = mb_convert_encoding($str, "EUC-JP", "auto");
?>
```

See also **mb_detect_order()**.

mb_convert_kana

(PHP 4 >= 4.0.6, PHP 5)

mb_convert_kana -- Convert "kana" one from another ("zen-kaku", "han-kaku" and more)

Description

string **mb_convert_kana** (string str [, string option [, string encoding]])

mb_convert_kana() performs "han-kaku" - "zen-kaku" conversion for string *str*. It returns converted string. This function is only useful for Japanese.

option is conversion option. Default value is *"KV"*.

encoding is character encoding. If it is omitted, internal character encoding is used.

Specify with combination of following options. Default value is *KV*.

Tabla 1. Applicable Conversion Options

Option	Meaning
r	Convert "zen-kaku" alphabets to "han-kaku"
R	Convert "han-kaku" alphabets to "zen-kaku"
n	Convert "zen-kaku" numbers to "han-kaku"
N	Convert "han-kaku" numbers to "zen-kaku"
a	Convert "zen-kaku" alphabets and numbers to "han-kaku"
A	Convert "han-kaku" alphabets and numbers to "zen-kaku" (Characters included in "a", "A" options are U+0021 - U+007E excluding U+0022, U+0027, U+005C, U+007E)
s	Convert "zen-kaku" space to "han-kaku" (U+3000 -> U+0020)
S	Convert "han-kaku" space to "zen-kaku" (U+0020 -> U+3000)
k	Convert "zen-kaku kata-kana" to "han-kaku kata-kana"
K	Convert "han-kaku kata-kana" to "zen-kaku kata-kana"
h	Convert "zen-kaku hira-gana" to "han-kaku kata-kana"

Option	Meaning
H	Convert "han-kaku kata-kana" to "zen-kaku hira-gana"
c	Convert "zen-kaku kata-kana" to "zen-kaku hira-gana"
C	Convert "zen-kaku hira-gana" to "zen-kaku kata-kana"
V	Collapse voiced sound notation and convert them into a character. Use with "K","H"

Ejemplo 1. mb_convert_kana() example

```
<?php
/* Convert all "kana" to "zen-kaku" "kata-kana" */
$str = mb_convert_kana($str, "KVC");

/* Convert "han-kaku" "kata-kana" to "zen-kaku" "kata-kana"
   and "zen-kaku" alpha-numeric to "han-kaku" */
$str = mb_convert_kana($str, "KVa");
?>
```

mb_convert_variables

(PHP 4 >= 4.0.6, PHP 5)

mb_convert_variables -- Convert character code in variable(s)

Description

string **mb_convert_variables** (string to_encoding, mixed from_encoding, mixed &vars [, mixed &...])

mb_convert_variables() convert character encoding of variables *vars* in encoding *from_encoding* to encoding *to_encoding*. It returns character encoding before conversion for success, **FALSE** for failure.

mb_convert_variables() join strings in Array or Object to detect encoding, since encoding detection tends to fail for short strings. Therefore, it is impossible to mix encoding in single array or object.

It *from_encoding* is specified by array or comma separated string, it tries to detect encoding from *from-coding*. When *encoding* is omitted, *detect_order* is used.

vars (3rd and larger) is reference to variable to be converted. String, Array and Object are accepted. **mb_convert_variables()** assumes all parameters have the same encoding.

Ejemplo 1. mb_convert_variables() example

```
<?php
/* Convert variables $post1, $post2 to internal encoding */
$interenc = mb_internal_encoding();
$inputenc = mb_convert_variables($interenc, "ASCII,UTF-8,SJIS-win", $post1, $post2);
?>
```

mb_decode_mimeheader

(PHP 4 >= 4.0.6, PHP 5)

mb_decode_mimeheader -- Decode string in MIME header field

Description

string **mb_decode_mimeheader** (string str)

mb_decode_mimeheader() decodes encoded-word string *str* in MIME header.

It returns decoded string in internal character encoding.

See also **mb_encode_mimeheader()**.

mb_decode_numericentity

(PHP 4 >= 4.0.6, PHP 5)

mb_decode_numericentity -- Decode HTML numeric string reference to character

Description

string **mb_decode_numericentity** (string str, array convmap [, string encoding])

Convert numeric string reference of string *str* in specified block to character. It returns converted string.

convmap is array to specifies code area to convert.

encoding is character encoding. If it is omitted, internal character encoding is used.

Ejemplo 1. *convmap* example

```
$convmap = array (
   int start_code1, int end_code1, int offset1, int mask1,
   int start_code2, int end_code2, int offset2, int mask2,
   ........
   int start_codeN, int end_codeN, int offsetN, int maskN );
// Specify Unicode value for start_codeN and end_codeN
// Add offsetN to value and take bit-wise 'AND' with maskN,
// then convert value to numeric string reference.
```

See also **mb_encode_numericentity()**.

mb_detect_encoding

(PHP 4 >= 4.0.6, PHP 5)

mb_detect_encoding -- Detect character encoding

Description

string **mb_detect_encoding** (string str [, mixed encoding_list [, bool strict]])

mb_detect_encoding() detects character encoding in string *str*. It returns detected character encoding.

encoding_list is list of character encoding. Encoding order may be specified by array or comma separated list string.

If *encoding_list* is omitted, detect_order is used.

Ejemplo 1. mb_detect_encoding() example

```
<?php
/* Detect character encoding with current detect_order */
echo mb_detect_encoding($str);

/* "auto" is expanded to "ASCII,JIS,UTF-8,EUC-JP,SJIS" */
echo mb_detect_encoding($str, "auto");

/* Specify encoding_list character encoding by comma separated list */
echo mb_detect_encoding($str, "JIS, eucjp-win, sjis-win");

/* Use array to specify encoding_list  */
$ary[] = "ASCII";
$ary[] = "JIS";
$ary[] = "EUC-JP";
echo mb_detect_encoding($str, $ary);
?>
```

See also **mb_detect_order()**.

mb_detect_order

(PHP 4 >= 4.0.6, PHP 5)

mb_detect_order -- Set/Get character encoding detection order

Description

array **mb_detect_order** ([mixed encoding_list])

mb_detect_order() sets automatic character encoding detection order to *encoding_list*. It returns **TRUE** for success, **FALSE** for failure.

encoding_list is array or comma separated list of character encoding. ("auto" is expanded to "ASCII, JIS, UTF-8, EUC-JP, SJIS")

If *encoding_list* is omitted, it returns current character encoding detection order as array.

This setting affects **mb_detect_encoding()** and **mb_send_mail()**.

> **Nota:** *mbstring* currently implements following encoding detection filters. If there is an invalid byte sequence for following encoding, encoding detection will fail.
>
> **Nota:** *UTF-8*, *UTF-7*, *ASCII*, *EUC-JP*,*SJIS*, *eucJP-win*, *SJIS-win*, *JIS*, *ISO-2022-JP*
>
> For *ISO-8859-**, *mbstring* always detects as *ISO-8859-**.
>
> For *UTF-16*, *UTF-32*, *UCS2* and *UCS4*, encoding detection will fail always.
>
> **Ejemplo 1. Useless detect order example**

```
; Always detect as ISO-8859-1
detect_order = ISO-8859-1, UTF-8

; Always detect as UTF-8, since ASCII/UTF-7 values are
; valid for UTF-8
detect_order = UTF-8, ASCII, UTF-7
```

Ejemplo 2. mb_detect_order() examples

```
<?php
/* Set detection order by enumerated list */
mb_detect_order("eucjp-win,sjis-win,UTF-8");

/* Set detection order by array */
$ary[] = "ASCII";
$ary[] = "JIS";
$ary[] = "EUC-JP";
mb_detect_order($ary);

/* Display current detection order */
echo implode(", ", mb_detect_order());
?>
```

See also **mb_internal_encoding()**, **mb_http_input()**, **mb_http_output()** and **mb_send_mail()**.

mb_encode_mimeheader

(PHP 4 >= 4.0.6, PHP 5)

mb_encode_mimeheader -- Encode string for MIME header

Description

string **mb_encode_mimeheader** (string str [, string charset [, string transfer_encoding [, string linefeed]]])

mb_encode_mimeheader() encodes a given string *str* by the MIME header encoding scheme. Returns a converted version of the string represented in ASCII.

charset specifies the name of the character set in which *str* is represented in. The default value is determined by the current NLS setting (*mbstring.language*).

transfer_encoding specifies the scheme of MIME encoding. It should be either *"B"* (Base64) or *"Q"* (Quoted-Printable). Falls back to *"B"* if not given.

linefeed specifies the EOL (end-of-line) marker with which **mb_encode_mime_header()** performs line-folding (a RFC term, the act of breaking a line longer than a certain length into multiple lines. The length is currently hard-coded to 74 characters). Falls back to *"\r\n"* (CRLF) if not given.

Ejemplo 1. mb_encode_mimeheader() example

```
<?php
$name = ""; // kanji
$mbox = "kru";
$doma = "gtinn.mon";
$addr = mb_encode_mimeheader($name, "UTF-7", "Q") . " <" . $mbox . "@" . $doma . ">";
echo $addr;
?>
```

> **Nota:** This function isn't designed to break lines at higher-level contextual break points (word boundaries, etc.). This behaviour may clutter up the original string with unexpected spaces.

See also **mb_decode_mimeheader()**.

mb_encode_numericentity

(PHP 4 >= 4.0.6, PHP 5)

mb_encode_numericentity -- Encode character to HTML numeric string reference

Description

string **mb_encode_numericentity** (string str, array convmap [, string encoding])

mb_encode_numericentity() converts specified character codes in string *str* from HTML numeric character reference to character code. It returns converted string.

convmap is array specifies code area to convert.

encoding is character encoding.

Ejemplo 1. *convmap* example

```
$convmap = array (
 int start_code1, int end_code1, int offset1, int mask1,
 int start_code2, int end_code2, int offset2, int mask2,
 ........
 int start_codeN, int end_codeN, int offsetN, int maskN );
// Specify Unicode value for start_codeN and end_codeN
// Add offsetN to value and take bit-wise 'AND' with maskN, then
// it converts value to numeric string reference.
```

Ejemplo 2. mb_encode_numericentity() example

```
<?php
/* Convert Left side of ISO-8859-1 to HTML numeric character reference */
$convmap = array(0x80, 0xff, 0, 0xff);
$str = mb_encode_numericentity($str, $convmap, "ISO-8859-1");

/* Convert user defined SJIS-win code in block 95-104 to numeric
   string reference */
$convmap = array(
       0xe000, 0xe03e, 0x1040, 0xffff,
       0xe03f, 0xe0bb, 0x1041, 0xffff,
       0xe0bc, 0xe0fa, 0x1084, 0xffff,
       0xe0fb, 0xe177, 0x1085, 0xffff,
       0xe178, 0xe1b6, 0x10c8, 0xffff,
       0xe1b7, 0xe233, 0x10c9, 0xffff,
       0xe234, 0xe272, 0x110c, 0xffff,
       0xe273, 0xe2ef, 0x110d, 0xffff,
       0xe2f0, 0xe32e, 0x1150, 0xffff,
       0xe32f, 0xe3ab, 0x1151, 0xffff );
$str = mb_encode_numericentity($str, $convmap, "sjis-win");
?>
```

See also **mb_decode_numericentity()**.

mb_ereg_match

(PHP 4 >= 4.2.0)

mb_ereg_match -- Regular expression match for multibyte string

Description

bool **mb_ereg_match** (string pattern, string string [, string option])

mb_ereg_match() returns **TRUE** if *string* matches regular expression *pattern*, **FALSE** if not.

The internal encoding or the character encoding specified in **mb_regex_encoding()** will be used as character encoding.

See also: **mb_regex_encoding()**, **mb_ereg()**.

mb_ereg_replace

(PHP 4 >= 4.2.0)

mb_ereg_replace -- Replace regular expression with multibyte support

Description

string **mb_ereg_replace** (string pattern, string replacement, string string [, array option])

mb_ereg_replace() scans *string* for matches to *pattern*, then replaces the matched text with *replacement* and returns the result string or **FALSE** on error. Multibyte character can be used in *pattern*.

Matching condition can be set by *option* parameter. If *i* is specified for this parameter, the case will be ignored. If *x* is specified, white space will be ignored. If *m* is specified, match will be executed in multiline mode and line break will be included in '.'. If *p* is specified, match will be executed in POSIX mode, line break will be considered as normal character. If *e* is specified, *replacement* string will be evaluated as PHP expression.

The internal encoding or the character encoding specified in **mb_regex_encoding()** will be used as character encoding.

See also: **mb_regex_encoding()**, **mb_eregi_replace()**.

mb_ereg_search_getpos

(PHP 4 >= 4.2.0)

mb_ereg_search_getpos -- Returns start point for next regular expression match

Description

array **mb_ereg_search_getpos** (void)

mb_ereg_search_getpos() returns the point to start regular expression match for **mb_ereg_search()**, **mb_ereg_search_pos()**, **mb_ereg_search_regs()**. The position is represented by bytes from the head of string.

The internal encoding or the character encoding specified in **mb_regex_encoding()** will be used as character encoding.

See also: **mb_regex_encoding()**, **mb_ereg_search_setpos()**.

mb_ereg_search_getregs

(PHP 4 >= 4.2.0)

mb_ereg_search_getregs -- Retrieve the result from the last multibyte regular expression match

Description

array **mb_ereg_search_getregs** (void)

mb_ereg_search_getregs() returns an array including the sub-string of matched part by last **mb_ereg_search()**, **mb_ereg_search_pos()**, **mb_ereg_search_regs()**. If there are some maches, the first element will have the matched sub-string, the second element will have the first part grouped with brackets, the third element will have the second part grouped with brackets, and so on. It returns **FALSE** on error;

The internal encoding or the character encoding specified in **mb_regex_encoding()** will be used as character encoding.

See also: **mb_regex_encoding()**, **mb_ereg_search_init()**.

mb_ereg_search_init

(PHP 4 >= 4.2.0)

mb_ereg_search_init -- Setup string and regular expression for multibyte regular expression match

Description

array **mb_ereg_search_init** (string string [, string pattern [, string option]])

mb_ereg_search_init() sets *string* and *pattern* for multibyte regular expression. These values are used for **mb_ereg_search()**, **mb_ereg_search_pos()**, **mb_ereg_search_regs()**. It returns **TRUE** for success, **FALSE** for error.

The internal encoding or the character encoding specified in **mb_regex_encoding()** will be used as character encoding.

See also: **mb_regex_encoding()**, **mb_ereg_search_regs()**.

mb_ereg_search_pos

(PHP 4 >= 4.2.0)

mb_ereg_search_pos -- Return position and length of matched part of multibyte regular expression

for predefined multibyte string

Description

array **mb_ereg_search_pos** ([string pattern [, string option]])

mb_ereg_search_pos() returns an array including position of matched part for multibyte regular expression. The first element of the array will be the beginning of matched part, the second element will be length (bytes) of matched part. It returns **FALSE** on error.

The string for match is specified by **mb_ereg_search_init()**. It it is not specified, the previous one will be used.

The internal encoding or the character encoding specified in **mb_regex_encoding()** will be used as character encoding.

See also: **mb_regex_encoding()**, **mb_ereg_search_init()**.

mb_ereg_search_regs

(PHP 4 >= 4.2.0)

mb_ereg_search_regs -- Returns the matched part of multibyte regular expression

Description

array **mb_ereg_search_regs** ([string pattern [, string option]])

mb_ereg_search_regs() executes the multibyte regular expression match, and if there are some matched part, it returns an array including substring of matched part as first element, the first grouped part with brackets as second element, the second grouped part as third element, and so on. It returns **FALSE** on error.

The internal encoding or the character encoding specified in **mb_regex_encoding()** will be used as character encoding.

See also: **mb_regex_encoding()**, **mb_ereg_search_init()**.

mb_ereg_search_setpos

(PHP 4 >= 4.2.0)

mb_ereg_search_setpos -- Set start point of next regular expression match

Description

array **mb_ereg_search_setpos** (int position)

mb_ereg_search_setpos() sets the starting point of match for **mb_ereg_search()**.

The internal encoding or the character encoding specified in **mb_regex_encoding()** will be used as character encoding.

See also: **mb_regex_encoding()**, **mb_ereg_search_init()**.

mb_ereg_search

(PHP 4 >= 4.2.0)

mb_ereg_search -- Multibyte regular expression match for predefined multibyte string

Description

bool **mb_ereg_search** ([string pattern [, string option]])

mb_ereg_search() returns **TRUE** if the multibyte string matches with the regular expression, **FALSE** for otherwise. The string for matching is set by **mb_ereg_search_init()**. If *pattern* is not specified, the previous one is used.

The internal encoding or the character encoding specified in **mb_regex_encoding()** will be used as character encoding.

See also: **mb_regex_encoding()**, **mb_ereg_search_init()**.

mb_ereg

(PHP 4 >= 4.2.0)

mb_ereg -- Regular expression match with multibyte support

Description

int **mb_ereg** (string pattern, string string [, array regs])

mb_ereg() executes the regular expression match with multibyte support, and returns 1 if matches are found. If the optional third parameter was specified, the function returns the byte length of matched part, and the array *regs* will contain the substring of matched string. The functions returns 1 if it matches with the empty string. If no matches found or error happend, **FALSE** will be returned.

The internal encoding or the character encoding specified in **mb_regex_encoding()** will be used as character encoding.

See also: **mb_regex_encoding()**, **mb_eregi()**

mb_eregi_replace

(PHP 4 >= 4.2.0)

mb_eregi_replace -- Replace regular expression with multibyte support ignoring case

Description

string **mb_eregi_replace** (string pattern, string replace, string string)

mb_ereg_replace() scans *string* for matches to *pattern*, then replaces the matched text with *replacement* and returns the result string or **FALSE** on error. Multibyte character can be used in *pattern*. The case will be ignored.

The internal encoding or the character encoding specified in **mb_regex_encoding()** will be used as character encoding.

See also: **mb_regex_encoding()**, **mb_ereg_replace()**.

mb_eregi

(PHP 4 >= 4.2.0)

mb_eregi -- Regular expression match ignoring case with multibyte support

Description

int **mb_eregi** (string pattern, string string [, array regs])

mb_eregi() executes the regular expression match with multibyte support, and returns 1 if matches are found. This function ignore case. If the optional third parameter was specified, the function returns the byte length of matched part, and the array *regs* will contain the substring of matched string. The functions returns 1 if it matches with the empty string. If no matches found or error happend, **FALSE** will be returned.

The internal encoding or the character encoding specified in **mb_regex_encoding()** will be used as character encoding.

See also: **mb_regex_encoding()**, **mb_ereg()**.

mb_get_info

(PHP 4 >= 4.2.0, PHP 5)

mb_get_info -- Get internal settings of mbstring

Description

string **mb_get_info** ([string type])

mb_get_info() returns internal setting parameter of mbstring.

If *type* isn't specified or is specified to "all", an array having the elements "internal_encoding", "http_output", "http_input", "func_overload" will be returned.

If *type* is specified for "http_output", "http_input", "internal_encoding", "func_overload", the

specified setting parameter will be returned.

See also **mb_internal_encoding()**, **mb_http_output()**.

mb_http_input

(PHP 4 >= 4.0.6, PHP 5)

mb_http_input -- Detect HTTP input character encoding

Description

string **mb_http_input** ([string type])

mb_http_input() returns result of HTTP input character encoding detection.

type: Input string specifies input type. "G" for GET, "P" for POST, "C" for COOKIE. If type is omitted, it returns last input type processed.

Return Value: Character encoding name. If **mb_http_input()** does not process specified HTTP input, it returns **FALSE**.

See also **mb_internal_encoding()**, **mb_http_output()**, **mb_detect_order()**.

mb_http_output

(PHP 4 >= 4.0.6, PHP 5)

mb_http_output -- Set/Get HTTP output character encoding

Description

string **mb_http_output** ([string encoding])

If *encoding* is set, **mb_http_output()** sets HTTP output character encoding to *encoding*. Output after this function is converted to *encoding*. **mb_http_output()** returns **TRUE** for success and **FALSE** for failure.

If *encoding* is omitted, **mb_http_output()** returns current HTTP output character encoding.

See also **mb_internal_encoding()**, **mb_http_input()**, **mb_detect_order()**.

mb_internal_encoding

(PHP 4 >= 4.0.6, PHP 5)

mb_internal_encoding -- Set/Get internal character encoding

Description

mixed **mb_internal_encoding** ([string encoding])

mb_internal_encoding() sets internal character encoding to *encoding* If parameter is omitted, it returns current internal encoding.

encoding is used for HTTP input character encoding conversion, HTTP output character encoding conversion and default character encoding for string functions defined by mbstring module.

encoding: Character encoding name

Return Value: If *encoding* is set,**mb_internal_encoding()** returns **TRUE** for success, otherwise returns **FALSE**. If *encoding* is omitted, it returns current character encoding name.

Ejemplo 1. mb_internal_encoding() example

```
<?php
/* Set internal character encoding to UTF-8 */
mb_internal_encoding("UTF-8");

/* Display current internal character encoding */
echo mb_internal_encoding();
?>
```

See also **mb_http_input()**, **mb_http_output()** and **mb_detect_order()**.

mb_language

(PHP 4 >= 4.0.6, PHP 5)

mb_language -- Set/Get current language

Description

string **mb_language** ([string language])

mb_language() sets language. If *language* is omitted, it returns current language as string.

language setting is used for encoding e-mail messages. Valid languages are "Japanese", "ja","English","en" and "uni" (UTF-8). **mb_send_mail()** uses this setting to encode e-mail.

Language and its setting is ISO-2022-JP/Base64 for Japanese, UTF-8/Base64 for uni, ISO-8859-1/quoted printable for English.

Return Value: If *language* is set and *language* is valid, it returns **TRUE**. Otherwise, it returns **FALSE**. When *language* is omitted, it returns language name as string. If no language is set previously, it returns **FALSE**.

See also **mb_send_mail()**.

mb_list_encodings

(PHP 5)

mb_list_encodings -- Returns an array of all supported encodings

Description

array **mb_list_encodings** (void)

mb_list_encodings() returns an array with all supported encodings.

Ejemplo 1. mb_list_encodings() example

```
<?php

print_r(mb_list_encodings());

?>
```

The above example will output:

```
Array
(
    [0] => pass
    [1] => auto
    [2] => wchar
    [3] => byte2be
    [4] => byte2le
    [5] => byte4be
    [6] => byte4le
    [7] => BASE64
    [8] => UUENCODE
    [9] => HTML-ENTITIES
    [10] => Quoted-Printable
    [11] => 7bit
    [12] => 8bit
    [13] => UCS-4
    [14] => UCS-4BE
    [15] => UCS-4LE
    [16] => UCS-2
    [17] => UCS-2BE
    [18] => UCS-2LE
    [19] => UTF-32
    [20] => UTF-32BE
    [21] => UTF-32LE
    [22] => UTF-16
    [23] => UTF-16BE
    [24] => UTF-16LE
    [25] => UTF-8
    [26] => UTF-7
    [27] => UTF7-IMAP
    [28] => ASCII
    [29] => EUC-JP
    [30] => SJIS
    [31] => eucJP-win
    [32] => SJIS-win
    [33] => JIS
    [34] => ISO-2022-JP
    [35] => Windows-1252
    [36] => ISO-8859-1
    [37] => ISO-8859-2
    [38] => ISO-8859-3
    [39] => ISO-8859-4
    [40] => ISO-8859-5
    [41] => ISO-8859-6
    [42] => ISO-8859-7
    [43] => ISO-8859-8
    [44] => ISO-8859-9
    [45] => ISO-8859-10
    [46] => ISO-8859-13
    [47] => ISO-8859-14
    [48] => ISO-8859-15
    [49] => EUC-CN
    [50] => CP936
    [51] => HZ
    [52] => EUC-TW
    [53] => BIG-5
    [54] => EUC-KR
    [55] => UHC
    [56] => ISO-2022-KR
    [57] => Windows-1251
    [58] => CP866
    [59] => KOI8-R
)
```

mb_output_handler

(PHP 4 >= 4.0.6, PHP 5)

mb_output_handler -- Callback function converts character encoding in output buffer

Description

string **mb_output_handler** (string contents, int status)

mb_output_handler() is **ob_start()** callback function. **mb_output_handler()** converts characters in output buffer from internal character encoding to HTTP output character encoding.

4.1.0 or later version, this handler adds charset HTTP header when following conditions are met:

- Does not set *Content-Type* by header()
- Default MIME type begins with *text/*
- *http_output* setting is other than pass

contents : Output buffer contents

status : Output buffer status

Return Value: String converted

Ejemplo 1. mb_output_handler() example

```
<?php
mb_http_output("UTF-8");
ob_start("mb_output_handler");
?>
```

> **Nota:** If you want to output some binary data such as image from PHP script with PHP 4.3.0 or later, Content-Type: header must be send using **header()** before any binary data was send to client (e.g. header("Content-Type: image/png")). If Content-Type: header was send, output character encoding conversion will not be performed.
>
> Note that if 'Content-Type: text/*' was send using **header()**, the sending data is regarded as text, encoding conversion will be performed using character encoding settings.
>
> If you want to output some binary data such as image from PHP script with PHP 4.2.x or earlier, you must set output encoding to "pass" using **mb_http_output()**.

See also **ob_start()**.

mb_parse_str

(PHP 4 >= 4.0.6, PHP 5)

mb_parse_str -- Parse GET/POST/COOKIE data and set global variable

Description

bool **mb_parse_str** (string encoded_string [, array &result])

mb_parse_str() parses GET/POST/COOKIE data and sets global variables. Since PHP does not

provide raw POST/COOKIE data, it can only used for GET data for now. It preses URL encoded data, detects encoding, converts coding to internal encoding and set values to *result* array or global variables.

encoded_string: URL encoded data.

result: Array contains decoded and character encoding converted values.

Return Value: It returns **TRUE** for success or **FALSE** for failure.

See also **mb_detect_order()**, **mb_internal_encoding()**.

mb_preferred_mime_name

(PHP 4 >= 4.0.6, PHP 5)

mb_preferred_mime_name -- Get MIME charset string

Description

string **mb_preferred_mime_name** (string encoding)

mb_preferred_mime_name() returns MIME *charset* string for character encoding *encoding*. It returns *charset* string.

Ejemplo 1. mb_preferred_mime_string() example

```
<?php
$outputenc = "sjis-win";
mb_http_output($outputenc);
ob_start("mb_output_handler");
header("Content-Type: text/html; charset=" . mb_preferred_mime_name($outputenc));
?>
```

mb_regex_encoding

(PHP 4 >= 4.2.0)

mb_regex_encoding -- Returns current encoding for multibyte regex as string

Description

string **mb_regex_encoding** ([string encoding])

mb_regex_encoding() returns the character encoding used by multibyte regex functions.

If the optional parameter *encoding* is specified, it is set to the character encoding for multibyte regex. The default value is the internal character encoding.

See also: **mb_internal_encoding()**, **mb_ereg()**

mb_regex_set_options

(PHP 4 >= 4.3.0)

mb_regex_set_options -- Set/Get the default options for mbregex functions

Description

string **mb_regex_set_options** ([string options])

mb_regex_set_options() sets the default options described by *options* for multibyte regex functions.

Returns the previous options. If *options* is omitted, it returns the string that describes the current options.

See also **mb_split()**, **mb_ereg()** and **mb_eregi()**

mb_send_mail

(PHP 4 >= 4.0.6, PHP 5)

mb_send_mail -- Send encoded mail

Description

bool **mb_send_mail** (string to, string subject, string message [, string additional_headers [, string additional_parameter]])

mb_send_mail() sends email. Headers and message are converted and encoded according to **mb_language()** setting. **mb_send_mail()** is wrapper function of **mail()**. See **mail()** for details.

to is mail addresses send to. Multiple recipients can be specified by putting a comma between each address in to. This parameter is not automatically encoded.

subject is subject of mail.

message is mail message.

additional_headers is inserted at the end of the header. This is typically used to add extra headers. Multiple extra headers are separated with a newline ("\n").

additional_parameter is a MTA command line parameter. It is useful when setting the correct Return-Path header when using sendmail.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

See also **mail()**, **mb_encode_mimeheader()**, and **mb_language()**.

mb_split

(PHP 4 >= 4.2.0)

mb_split -- Split multibyte string using regular expression

Description

array **mb_split** (string pattern, string string [, int limit])

mb_split() split multibyte *string* using regular expression *pattern* and returns the result as an array.

If optional parameter *limit* is specified, it will be split in *limit* elements as maximum.

The internal encoding or the character encoding specified in **mb_regex_encoding()** will be used as character encoding.

See also: **mb_regex_encoding()**, **mb_ereg()**.

mb_strcut

(PHP 4 >= 4.0.6, PHP 5)

mb_strcut -- Get part of string

Description

string **mb_strcut** (string str, int start [, int length [, string encoding]])

mb_strcut() returns the portion of *str* specified by the *start* and *length* parameters.

mb_strcut() performs ñalent operation as **mb_substr()** with different method. If *start* position is multi-byte character's second byte or larger, it starts from first byte of multi-byte character.

It subtracts string from *str* that is shorter than *length* AND character that is not part of multi-byte string or not being middle of shift sequence.

encoding is character encoding. If it is not set, internal character encoding is used.

See also **mb_substr()**, **mb_internal_encoding()**.

mb_strimwidth

(PHP 4 >= 4.0.6, PHP 5)

mb_strimwidth -- Get truncated string with specified width

Description

string **mb_strimwidth** (string str, int start, int width [, string trimmarker [, string encoding]])

mb_strimwidth() truncates string *str* to specified *width*. It returns truncated string.

If *trimmarker* is set, *trimmarker* is appended to return value.

start is start position offset. Number of characters from the beginning of string. (First character is 0)

trimmarker is string that is added to the end of string when string is truncated.

encoding is character encoding. If it is omitted, internal encoding is used.

Ejemplo 1. mb_strimwidth() example

```
<?php
$str = mb_strimwidth($str, 0, 40, "..>");
?>
```

See also **mb_strwidth()** and **mb_internal_encoding()**.

mb_strlen

(PHP 4 >= 4.0.6, PHP 5)

mb_strlen -- Get string length

Description

string **mb_strlen** (string str [, string encoding])

mb_strlen() returns number of characters in string *str* having character encoding *encoding*. A multi-byte character is counted as 1.

encoding is character encoding for *str*. If *encoding* is omitted, internal character encoding is used.

See also **mb_internal_encoding()**, **strlen()**.

mb_strpos

(PHP 4 >= 4.0.6, PHP 5)

mb_strpos -- Find position of first occurrence of string in a string

Description

int **mb_strpos** (string haystack, string needle [, int offset [, string encoding]])

mb_strpos() returns the numeric position of the first occurrence of *needle* in the *haystack* string. If *needle* is not found, it returns **FALSE**.

mb_strpos() performs multi-byte safe **strpos()** operation based on number of characters. *needle* position is counted from the beginning of the *haystack*. First character's position is 0. Second character position is 1, and so on.

If *encoding* is omitted, internal character encoding is used. **mb_strrpos()** accepts *string* for *needle* where **strrpos()** accepts only character.

offset is search offset. If it is not specified, 0 is used.

encoding is character encoding name. If it is omitted, internal character encoding is used.

See also **mb_strpos()**, **mb_internal_encoding()**, **strpos()**

mb_strrpos

(PHP 4 >= 4.0.6, PHP 5)

mb_strrpos -- Find position of last occurrence of a string in a string

Description

int **mb_strrpos** (string haystack, string needle [, string encoding])

mb_strrpos() returns the numeric position of the last occurrence of *needle* in the *haystack* string. If *needle* is not found, it returns **FALSE**.

mb_strrpos() performs multi-byte safe **strrpos()** operation based on number of characters. *needle* position is counted from the beginning of *haystack*. First character's position is 0. Second character position is 1.

If *encoding* is omitted, internal encoding is assumed. **mb_strrpos()** accepts *string* for *needle* where **strrpos()** accepts only character.

encoding is character encoding. If it is not specified, internal character encoding is used.

See also **mb_strpos()**, **mb_internal_encoding()**, **strrpos()**.

mb_strtolower

(PHP 4 >= 4.3.0, PHP 5)

mb_strtolower -- Make a string lowercase

Description

string **mb_strtolower** (string str [, string encoding])

mb_strtolower() returns *str* with all alphabetic characters converted to lowercase.

encoding specifies the encoding of *str*; if omitted, the internal character encoding value will be used.

For more information about the Unicode properties, please see http://www.unicode.org/unicode/reports/tr21/.

By contrast to **strtolower()**, 'alphabetic' is determined by the Unicode character properties. Thus the behaviour of this function is not affected by locale settings and it can convert any characters that have 'alphabetic' property, such as A-umlaut (Ãƒâ€ž).

Ejemplo 1. mb_strtolower() example

```
<?php
$str = "Mary Had A Little Lamb and She LOVED It So";
$str = mb_strtolower($str);
echo $str; // Prints mary had a little lamb and she loved it so
?>
```

See also **strtolower()**, **mb_strtoupper()** and **mb_convert_case()**.

mb_strtoupper

(PHP 4 >= 4.3.0, PHP 5)

mb_strtoupper -- Make a string uppercase

Description

string **mb_strtoupper** (string str [, string encoding])

mb_strtoupper() returns *str* with all alphabetic characters converted to uppercase.

encoding specifies the encoding of *str*; if omitted, the internal character encoding value will be used.

By contrast to **strtoupper()**, 'alphabetic' is determined by the Unicode character properties. Thus the behaviour of this function is not affected by locale settings and it can convert any characters that have 'alphabetic' property, such as a-umlaut (ÃƒÂ¤).

For more information about the Unicode properties, please see http://www.unicode.org/unicode/reports/tr21/.

Ejemplo 1. mb_strtoupper() example

```
<?php
$str = "Mary Had A Little Lamb and She LOVED It So";
$str = mb_strtoupper($str);
echo $str; // Prints MARY HAD A LITTLE LAMB AND SHE LOVED IT SO
?>
```

See also **strtoupper()**, **mb_strtolower()** and **mb_convert_case()**.

mb_strwidth

(PHP 4 >= 4.0.6, PHP 5)

mb_strwidth -- Return width of string

Description

int **mb_strwidth** (string str [, string encoding])

mb_strwidth() returns width of string *str*.

Multi-byte character usually twice of width compare to single byte character.

Tabla 1. Characters width

Chars	Width
U+0000 - U+0019	0
U+0020 - U+1FFF	1
U+2000 - U+FF60	2
U+FF61 - U+FF9F	1
U+FFA0 -	2

encoding is character encoding. If it is omitted, internal encoding is used.

See also: **mb_strimwidth()**, **mb_internal_encoding()**.

mb_substitute_character

(PHP 4 >= 4.0.6, PHP 5)

mb_substitute_character -- Set/Get substitution character

Description

mixed **mb_substitute_character** ([mixed substrchar])

mb_substitute_character() specifies substitution character when input character encoding is invalid or character code is not exist in output character encoding. Invalid characters may be substituted **NULL**(no output), string or integer value (Unicode character code value).

This setting affects **mb_convert_encoding()**, **mb_convert_variables()**, **mb_output_handler()**, and **mb_send_mail()**.

substchar : Specify Unicode value as integer or specify as string as follows

- "none" : no output
- "long" : Output character code value (Example: U+3000,JIS+7E7E)

Return Value: If *substchar* is set, it returns **TRUE** for success, otherwise returns **FALSE**. If *substchar* is not set, it returns Unicode value or "*none*"/"*long*".

Ejemplo 1. mb_substitute_character() example

```php
<?php
/* Set with Unicode U+3013 (GETA MARK) */
mb_substitute_character(0x3013);

/* Set hex format */
mb_substitute_character("long");

/* Display current setting */
echo mb_substitute_character();
?>
```

mb_substr_count

(PHP 4 >= 4.3.0, PHP 5)

mb_substr_count -- Count the number of substring occurrences

Description

int **mb_substr_count** (string haystack, string needle [, string encoding])

mb_substr_count() returns the number of times the *needle* substring occurs in the *haystack* string.

encoding specifies the encoding for *needle* and *haystack*. If omitted, internal character encoding is used.

Ejemplo 1. mb_substr_count() example

```php
<?php
echo mb_substr_count("This is a test", "is"); // prints out 2
?>
```

See also **substr_count()**, **mb_strpos()**, **mb_substr()**.

mb_substr

(PHP 4 >= 4.0.6, PHP 5)

mb_substr -- Get part of string

Description

string **mb_substr** (string str, int start [, int length [, string encoding]])

mb_substr() returns the portion of *str* specified by the *start* and *length* parameters.

mb_substr() performs multi-byte safe **substr()** operation based on number of characters. Position is counted from the beginning of *str*. First character's position is 0. Second character position is 1, and so on.

If *encoding* is omitted, internal encoding is assumed.

encoding is character encoding. If it is omitted, internal character encoding is used.

See also **mb_strcut()**, **mb_internal_encoding()**.

LXVI. MCAL functions

MCAL stands for Modular Calendar Access Library.

Libmcal is a C library for accessing calendars. It's written to be very modular, with plugable drivers. MCAL is the calendar ñalent of the IMAP module for mailboxes.

With mcal support, a calendar stream can be opened much like the mailbox stream with the IMAP support. Calendars can be local file stores, remote ICAP servers, or other formats that are supported by the mcal library.

Calendar events can be pulled up, queried, and stored. There is also support for calendar triggers (alarms) and reoccuring events.

With libmcal, central calendar servers can be accessed and used, removing the need for any specific database or local file programming.

To get these functions to work, you have to compile PHP with *--with-mcal*. That requires the mcal library to be installed. Grab the latest version from http://mcal.chek.com/ and compile and install it.

The following constants are defined when using the MCAL module: MCAL_SUNDAY, MCAL_MONDAY, MCAL_TUESDAY, MCAL_WEDNESDAY, MCAL_THURSDAY, MCAL_FRIDAY, MCAL_SATURDAY, MCAL_RECUR_NONE, MCAL_RECUR_DAILY, MCAL_RECUR_WEEKLY, MCAL_RECUR_MONTHLY_MDAY, MCAL_RECUR_MONTHLY_WDAY, MCAL_RECUR_YEARLY, MCAL_JANUARY, MCAL_FEBRUARY, MCAL_MARCH, MCAL_APRIL, MCAL_MAY, MCAL_JUNE, MCAL_JULY, MCAL_AUGUGT, MCAL_SEPTEMBER, MCAL_OCTOBER, MCAL_NOVEMBER, and MCAL_DECEMBER. Most of the functions use an internal event structure that is unique for each stream. This alleviates the need to pass around large objects between functions. There are convenience functions for setting, initializing, and retrieving the event structure values.

Tabla de contenidos

mcal_append_event

(PHP 4)

mcal_append_event -- Store a new event into an MCAL calendar

Description

int **mcal_append_event** (int mcal_stream)

mcal_append_event() Stores the global event into an MCAL calendar for the given stream.

Returns the uid of the newly inserted event.

mcal_close

(PHP 3>= 3.0.13, PHP 4)

mcal_close -- Close an MCAL stream

Description

int **mcal_close** (int mcal_stream, int flags)

Closes the given mcal stream.

mcal_create_calendar

(PHP 3>= 3.0.13, PHP 4)

mcal_create_calendar -- Create a new MCAL calendar

Description

bool **mcal_create_calendar** (int stream, string calendar)

Creates a new calendar named *calendar*.

mcal_date_compare

(PHP 3>= 3.0.13, PHP 4)

mcal_date_compare -- Compares two dates

Description

int **mcal_date_compare** (int a_year, int a_month, int a_day, int b_year, int b_month, int b_day)

mcal_date_compare() Compares the two given dates, returns <0, 0, >0 if a<b, a==b, a>b respectively

mcal_date_valid

(PHP 3>= 3.0.13, PHP 4)

mcal_date_valid -- Returns **TRUE** if the given year, month, day is a valid date

Description

int **mcal_date_valid** (int year, int month, int day)

mcal_date_valid() Returns **TRUE** if the given year, month and day is a valid date, **FALSE** if not.

mcal_day_of_week

(PHP 3>= 3.0.13, PHP 4)

mcal_day_of_week -- Returns the day of the week of the given date

Description

int **mcal_day_of_week** (int year, int month, int day)

mcal_day_of_week() returns the day of the week of the given date

mcal_day_of_year

(PHP 3>= 3.0.13, PHP 4)

mcal_day_of_year -- Returns the day of the year of the given date

Description

int **mcal_day_of_year** (int year, int month, int day)

mcal_day_of_year() returns the day of the year of the given date

mcal_days_in_month

(PHP 3>= 3.0.13, PHP 4)

mcal_days_in_month -- Returns the number of days in the given month

Description

int **mcal_days_in_month** (int month, int leap year)

mcal_days_in_month() Returns the number of days in the given month, taking into account if the given year is a leap year or not.

mcal_delete_calendar

(PHP 3>= 3.0.13, PHP 4)

mcal_delete_calendar -- Delete an MCAL calendar

Description

string **mcal_delete_calendar** (int stream, string calendar)

Deletes the calendar named *calendar*.

mcal_delete_event

(PHP 3>= 3.0.13, PHP 4)

mcal_delete_event -- Delete an event from an MCAL calendar

Description

int **mcal_delete_event** (int uid)

mcal_delete_event() deletes the calendar event specified by the uid.

Returns **TRUE**.

mcal_event_add_attribute

(PHP 3>= 3.0.15, PHP 4)

mcal_event_add_attribute -- Adds an attribute and a value to the streams global event structure

Description

void **mcal_event_add_attribute** (int stream, string attribute, string value)

mcal_event_add_attribute() adds an attribute to the stream's global event structure with the value given by "value" .

mcal_event_init

(PHP 3>= 3.0.13, PHP 4)

mcal_event_init -- Initializes a streams global event structure

Description

int **mcal_event_init** (int stream)

mcal_event_init() initializes a streams global event structure. this effectively sets all elements of the structure to 0, or the default settings.

Returns **TRUE**.

mcal_event_set_alarm

(PHP 3>= 3.0.13, PHP 4)

mcal_event_set_alarm -- Sets the alarm of the streams global event structure

Description

int **mcal_event_set_alarm** (int stream, int alarm)

mcal_event_set_alarm() sets the streams global event structure's alarm to the given minutes before the event.

Returns **TRUE**.

mcal_event_set_category

(PHP 3>= 3.0.13, PHP 4)

mcal_event_set_category -- Sets the category of the streams global event structure

Description

int **mcal_event_set_category** (int stream, string category)

mcal_event_set_category() sets the streams global event structure's category to the given string.

Returns **TRUE**.

mcal_event_set_class

(PHP 3>= 3.0.13, PHP 4)

mcal_event_set_class -- Sets the class of the streams global event structure

Description

int **mcal_event_set_class** (int stream, int class)

mcal_event_set_class() sets the streams global event structure's class to the given value. The class is either 1 for public, or 0 for private.

Returns **TRUE**.

mcal_event_set_description

(PHP 3>= 3.0.13, PHP 4)

mcal_event_set_description -- Sets the description of the streams global event structure

Description

int **mcal_event_set_description** (int stream, string description)

mcal_event_set_description() sets the streams global event structure's description to the given string.

Returns **TRUE**.

mcal_event_set_end

(PHP 3>= 3.0.13, PHP 4)

mcal_event_set_end -- Sets the end date and time of the streams global event structure

Description

int **mcal_event_set_end** (int stream, int year, int month [, int day [, int hour [, int min [, int sec]]]])

mcal_event_set_end() sets the streams global event structure's end date and time to the given values.

Returns **TRUE**.

mcal_event_set_recur_daily

(PHP 3>= 3.0.13, PHP 4)

mcal_event_set_recur_daily -- Sets the recurrence of the streams global event structure

Description

int **mcal_event_set_recur_daily** (int stream, int year, int month, int day, int interval)

mcal_event_set_recur_daily() sets the streams global event structure's recurrence to the given value to be reoccuring on a daily basis, ending at the given date.

mcal_event_set_recur_monthly_mday

(PHP 3>= 3.0.13, PHP 4)

mcal_event_set_recur_monthly_mday -- Sets the recurrence of the streams global event structure

Description

int **mcal_event_set_recur_monthly_mday** (int stream, int year, int month, int day, int interval)

mcal_event_set_recur_monthly_mday() sets the streams global event structure's recurrence to the given value to be reoccuring on a monthly by month day basis, ending at the given date.

mcal_event_set_recur_monthly_wday

(PHP 3>= 3.0.13, PHP 4)

mcal_event_set_recur_monthly_wday -- Sets the recurrence of the streams global event structure

Description

int **mcal_event_set_recur_monthly_wday** (int stream, int year, int month, int day, int interval)

mcal_event_set_recur_monthly_wday() sets the streams global event structure's recurrence to the given value to be reoccuring on a monthly by week basis, ending at the given date.

mcal_event_set_recur_none

(PHP 3>= 3.0.15, PHP 4)

mcal_event_set_recur_none -- Sets the recurrence of the streams global event structure

Description

int **mcal_event_set_recur_none** (int stream)

mcal_event_set_recur_none() sets the streams global event structure to not recur (event->recur_type is set to MCAL_RECUR_NONE).

mcal_event_set_recur_weekly

(PHP 3>= 3.0.13, PHP 4)

mcal_event_set_recur_weekly -- Sets the recurrence of the streams global event structure

Description

int **mcal_event_set_recur_weekly** (int stream, int year, int month, int day, int interval, int weekdays)

mcal_event_set_recur_weekly() sets the streams global event structure's recurrence to the given value to be reoccuring on a weekly basis, ending at the given date.

mcal_event_set_recur_yearly

(PHP 3>= 3.0.13, PHP 4)

mcal_event_set_recur_yearly -- Sets the recurrence of the streams global event structure

Description

int **mcal_event_set_recur_yearly** (int stream, int year, int month, int day, int interval)

mcal_event_set_recur_yearly() sets the streams global event structure's recurrence to the given value to be reoccuring on a yearly basis,ending at the given date .

mcal_event_set_start

(PHP 3>= 3.0.13, PHP 4)

mcal_event_set_start -- Sets the start date and time of the streams global event structure

Description

int **mcal_event_set_start** (int stream, int year, int month [, int day [, int hour [, int min [, int sec]]]])

mcal_event_set_start() sets the streams global event structure's start date and time to the given values.

Returns **TRUE**.

mcal_event_set_title

(PHP 3>= 3.0.13, PHP 4)

mcal_event_set_title -- Sets the title of the streams global event structure

Description

int **mcal_event_set_title** (int stream, string title)

mcal_event_set_title() sets the streams global event structure's title to the given string.

Returns **TRUE**.

mcal_expunge

(no version information, might be only in CVS)

mcal_expunge -- Deletes all events marked for being expunged

Description

int **mcal_expunge** (int stream)

mcal_expunge() deletes all events which have been previously marked for deletion.

mcal_fetch_current_stream_event

(PHP 3>= 3.0.13, PHP 4)

mcal_fetch_current_stream_event -- Returns an object containing the current streams event structure

Description

int **mcal_fetch_current_stream_event** (int stream)

mcal_event_fetch_current_stream_event() returns the current stream's event structure as an object containing:

- int id - ID of that event.
- int public - **TRUE** if the event if public, **FALSE** if it is private.
- string category - Category string of the event.
- string title - Title string of the event.

- string description - Description string of the event.
- int alarm - number of minutes before the event to send an alarm/reminder.
- object start - Object containing a datetime entry.
- object end - Object containing a datetime entry.
- int recur_type - recurrence type
- int recur_interval - recurrence interval
- datetime recur_enddate - recurrence end date
- int recur_data - recurrence data

All datetime entries consist of an object that contains:

- int year - year
- int month - month
- int mday - day of month
- int hour - hour
- int min - minutes
- int sec - seconds
- int alarm - minutes before event to send an alarm

mcal_fetch_event

(PHP 3>= 3.0.13, PHP 4)

mcal_fetch_event -- Fetches an event from the calendar stream

Description

object **mcal_fetch_event** (int mcal_stream, int event_id [, int options])

mcal_fetch_event() fetches an event from the calendar stream specified by *id*.

Returns an event object consisting of:

- int id - ID of that event.
- int public - **`TRUE`** if the event if public, **`FALSE`** if it is private.
- string category - Category string of the event.

- string title - Title string of the event.
- string description - Description string of the event.
- int alarm - number of minutes before the event to send an alarm/reminder.
- object start - Object containing a datetime entry.
- object end - Object containing a datetime entry.
- int recur_type - recurrence type
- int recur_interval - recurrence interval
- datetime recur_enddate - recurrence end date
- int recur_data - recurrence data

All datetime entries consist of an object that contains:

- int year - year
- int month - month
- int mday - day of month
- int hour - hour
- int min - minutes
- int sec - seconds
- int alarm - minutes before event to send an alarm

mcal_is_leap_year

(PHP 3>= 3.0.13, PHP 4)

mcal_is_leap_year -- Returns if the given year is a leap year or not

Description

int **mcal_is_leap_year** (int year)

mcal_is_leap_year() returns 1 if the given year is a leap year, 1 if not.

mcal_list_alarms

(PHP 3>= 3.0.13, PHP 4)

mcal_list_alarms -- Return a list of events that has an alarm triggered at the given datetime

Description

array **mcal_list_alarms** (int mcal_stream [, int begin_year [, int begin_month [, int begin_day [, int end_year [, int end_month [, int end_day]]]]]])

Returns an array of event ID's that has an alarm going off between the start and end dates, or if just a stream is given, uses the start and end dates in the global event structure.

mcal_list_events() function takes in an optional beginning date and an end date for a calendar stream. An array of event id's that are between the given dates or the internal event dates are returned.

mcal_list_events

(PHP 3>= 3.0.13, PHP 4)

mcal_list_events -- Return a list of events between two given datetimes

Description

array **mcal_list_events** (int mcal_stream [, int begin_year [, int begin_month [, int begin_day [, int end_year [, int end_month [, int end_day]]]]]])

Returns an array of event ID's that are between the start and end dates, or if just a stream is given, uses the start and end dates in the global event structure.

mcal_list_events() function takes in an optional beginning date and an end date for a calendar stream. An array of event id's that are between the given dates or the internal event dates are returned.

mcal_next_recurrence

(PHP 3>= 3.0.13, PHP 4)

mcal_next_recurrence -- Returns the next recurrence of the event

Description

int **mcal_next_recurrence** (int stream, int weekstart, array next)

mcal_next_recurrence() returns an object filled with the next date the event occurs, on or after the supplied date. Returns empty date field if event does not occur or something is invalid. Uses weekstart to determine what day is considered the beginning of the week.

mcal_open

(PHP 3>= 3.0.13, PHP 4)

mcal_open -- Opens up an MCAL connection

Description

int **mcal_open** (string calendar, string username, string password, string options)

Returns an MCAL stream on success, **FALSE** on error.

mcal_open() opens up an MCAL connection to the specified *calendar* store. If the optional *options* is specified, passes the *options* to that mailbox also. The streams internal event structure is also initialized upon connection.

mcal_popen

(PHP 3>= 3.0.13, PHP 4)

mcal_popen -- Opens up a persistent MCAL connection

Description

int **mcal_popen** (string calendar, string username, string password [, int options])

Returns an MCAL stream on success, **FALSE** on error.

mcal_popen() opens up an MCAL connection to the specified *calendar* store. If the optional *options* is specified, passes the *options* to that mailbox also. The streams internal event structure is also initialized upon connection.

mcal_rename_calendar

(PHP 3>= 3.0.13, PHP 4)

mcal_rename_calendar -- Rename an MCAL calendar

Description

string **mcal_rename_calendar** (int stream, string old_name, string new_name)

Renames the calendar *old_name* to *new_name*.

mcal_reopen

(PHP 3>= 3.0.13, PHP 4)

mcal_reopen -- Reopens an MCAL connection

Description

int **mcal_reopen** (int mcal_stream, string calendar [, int options])

Reopens an MCAL stream to a new calendar.

mcal_reopen() reopens an MCAL connection to the specified *calendar* store. If the optional *options* is specified, passes the *options* to that mailbox also.

mcal_snooze

(PHP 3>= 3.0.13, PHP 4)

mcal_snooze -- Turn off an alarm for an event

Description

int **mcal_snooze** (int uid)

mcal_snooze() turns off an alarm for a calendar event specified by the uid.

Returns **TRUE**.

mcal_store_event

(PHP 3>= 3.0.13, PHP 4)

mcal_store_event -- Modify an existing event in an MCAL calendar

Description

int **mcal_store_event** (int mcal_stream)

mcal_store_event() Stores the modifications to the current global event for the given stream.

Returns **TRUE** on success and **FALSE** on error.

mcal_time_valid

(PHP 3>= 3.0.13, PHP 4)

mcal_time_valid -- Returns **TRUE** if the given year, month, day is a valid time

Description

int **mcal_time_valid** (int hour, int minutes, int seconds)

mcal_time_valid() Returns **TRUE** if the given hour, minutes and seconds is a valid time, **FALSE** if not.

mcal_week_of_year

(PHP 4)

mcal_week_of_year -- Returns the week number of the given date

Description

int **mcal_week_of_year** (int day, int month, int year)

mcal_week_of_year() returns the week number of the given date.

LXVII. Funciones de Cifrado Mcrypt

Introducción

Esta es una interfaz con la biblioteca mcrypt, la cual soporta una gran variedad de algoritmos de bloque como DES, TripleDES, Blowfish (por defecto), 3-WAY, SAFER-SK64, SAFER-SK128, TWOFISH, TEA, RC2 y GOST en los modos de cifrado CBC, OFB, CFB y ECB. Adicionalmente, soporta RC6 e IDEA, los cuales se consideran "no-libres".

Requirimientos

Estas funciones trabajan usando mcrypt. Para usar la biblioteca, descargue `libmcrypt-x.x.tar.gz` desde http://mcrypt.sourceforge.net/ y siga las instrucciones de instalación incluidas. Los usuarios de windows encontrarán todos los binarios compilados de mcrypt que se necesitan en http://ftp.emini.dk/pub/php/win32/mcrypt/.

A partir de PHP 5.0.0, es necesario usar la versión 2.5.6 o superior de libmcrypt.

Si ha enlazado el software contra libmcrypt 2.4.x o versiones superiores, los siguientes algoritmos de bloque adicionales son soportados: CAST, LOKI97, RIJNDAEL, SAFERPLUS, SERPENT y los siguientes cifrados de secuencia: ENIGMA (cifrado), PANAMA, RC4 y WAKE. Con libmcrypt 2.4.x o superior, también se encuentra disponible otro modo de cifrado; nOFB.

Instalación

You need to compile PHP with the *--with-mcrypt[=DIR]* parameter to enable this extension. DIR is the mcrypt install directory. Make sure you compile libmcrypt with the option *--disable-posix-threads*.

Configuración en tiempo de ejecución

El comportamiento de estas funciones está afectado por los valores definidos en `php.ini`.

Tabla 1. Mcrypt configuration options

Name	Default	Changeable
mcrypt.algorithms_dir	NULL	PHP_INI_ALL
mcrypt.modes_dir	NULL	PHP_INI_ALL

For further details and definitions of the PHP_INI_* constants, see the Apéndice H.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

Mcrypt can operate in four block cipher modes (CBC, OFB, CFB, and ECB). If linked against libmcrypt-2.4.x or higher the functions can also operate in the block cipher mode nOFB and in STREAM mode. Below you find a list with all supported encryption modes together with the constants that are defines for the encryption mode. For a more complete reference and discussion see Applied Cryptography by Schneier (ISBN 0-471-11709-9).

- MCRYPT_MODE_ECB (electronic codebook) is suitable for random data, such as encrypting other keys. Since data there is short and random, the disadvantages of ECB have a favorable negative effect.
- MCRYPT_MODE_CBC (cipher block chaining) is especially suitable for encrypting files where the security is increased over ECB significantly.
- MCRYPT_MODE_CFB (cipher feedback) is the best mode for encrypting byte streams where single bytes must be encrypted.
- MCRYPT_MODE_OFB (output feedback, in 8bit) is comparable to CFB, but can be used in applications where error propagation cannot be tolerated. It's insecure (because it operates in 8bit mode) so it is not recommended to use it.
- MCRYPT_MODE_NOFB (output feedback, in nbit) is comparable to OFB, but more secure because it operates on the block size of the algorithm.
- MCRYPT_MODE_STREAM is an extra mode to include some stream algorithms like WAKE or RC4.

Some other mode and random device constants:

`MCRYPT_ENCRYPT` (**integer**)

`MCRYPT_DECRYPT` (**integer**)

MCRYPT_DEV_RANDOM (integer)

MCRYPT_DEV_URANDOM (integer)

MCRYPT_RAND (integer)

Cifrados mcrypt

Aquí se encuentra una lista de cifrados que son soportados actualmente por la extensión mcrypt. Para una lista completa de los cifrados soportados, consulte las definiciones al final de `mcrypt.h`. La regla general con la API de mcrypt-2.2.x API es que puede acceder al cifrado desde PHP mediante MCRYPT_nombre_cifrado. Con la API de libmcrypt-2.4.x y libmcrypt-2.5.x API estas constantes funcionan también, pero es posible especificar el nombre del cifrado como una cadena con una llamada a **mcrypt_module_open()**.

- MCRYPT_3DES
- MCRYPT_ARCFOUR_IV (libmcrypt > 2.4.x únicamente)
- MCRYPT_ARCFOUR (libmcrypt > 2.4.x únicamente)
- MCRYPT_BLOWFISH
- MCRYPT_CAST_128
- MCRYPT_CAST_256
- MCRYPT_CRYPT
- MCRYPT_DES
- MCRYPT_DES_COMPAT (libmcrypt 2.2.x únicamente)
- MCRYPT_ENIGMA (libmcrypt > 2.4.x únicamente, alias para MCRYPT_CRYPT)
- MCRYPT_GOST
- MCRYPT_IDEA (non-free)
- MCRYPT_LOKI97 (libmcrypt > 2.4.x únicamente)
- MCRYPT_MARS (libmcrypt > 2.4.x únicamente, no-libre)
- MCRYPT_PANAMA (libmcrypt > 2.4.x únicamente)
- MCRYPT_RIJNDAEL_128 (libmcrypt > 2.4.x únicamente)
- MCRYPT_RIJNDAEL_192 (libmcrypt > 2.4.x únicamente)
- MCRYPT_RIJNDAEL_256 (libmcrypt > 2.4.x únicamente)
- MCRYPT_RC2

- MCRYPT_RC4 (libmcrypt 2.2.x únicamente)
- MCRYPT_RC6 (libmcrypt > 2.4.x únicamente)
- MCRYPT_RC6_128 (libmcrypt 2.2.x únicamente)
- MCRYPT_RC6_192 (libmcrypt 2.2.x únicamente)
- MCRYPT_RC6_256 (libmcrypt 2.2.x únicamente)
- MCRYPT_SAFER64
- MCRYPT_SAFER128
- MCRYPT_SAFERPLUS (libmcrypt > 2.4.x únicamente)
- MCRYPT_SERPENT(libmcrypt > 2.4.x únicamente)
- MCRYPT_SERPENT_128 (libmcrypt 2.2.x únicamente)
- MCRYPT_SERPENT_192 (libmcrypt 2.2.x únicamente)
- MCRYPT_SERPENT_256 (libmcrypt 2.2.x únicamente)
- MCRYPT_SKIPJACK (libmcrypt > 2.4.x únicamente)
- MCRYPT_TEAN (libmcrypt 2.2.x únicamente)
- MCRYPT_THREEWAY
- MCRYPT_TRIPLEDES (libmcrypt > 2.4.x únicamente)
- MCRYPT_TWOFISH (para versinos mcrypt 2.x más antiguas, o mcrypt > 2.4.x)
- MCRYPT_TWOFISH128 (TWOFISHxxx se encuentran disponibles en versiones 2.x más recientes, pero no en las versiones 2.4.x)
- MCRYPT_TWOFISH192
- MCRYPT_TWOFISH256
- MCRYPT_WAKE (libmcrypt > 2.4.x únicamente)
- MCRYPT_XTEA (libmcrypt > 2.4.x únicamente)

Usted debe (en modos CFB y OFB) o puede (en modo CBC) entregar un vector de inicialización (IV, por sus siglas en Inglés) a la respectiva función de cifrado. El IV debe ser único y debe ser el mismo cuando se realice descifrado/cifrado. Con datos que sean almacenados en forma cifrada, usted puede tomar la salida de una función del índice bajo el cual se encuentren almacenados los datos (p.ej. la llave MD5 del nombre de archivo). Alternativamente, puede transmitir el IV junto con los datos cifrados (vea el capítulo 9.3 de Applied Cryptography by Schneier (ISBN 0-471-11709-9) para una discusión de este tópico).

Ejemplos

Mcrypt puede usarse para cifrar y descifrar usando los cifrados mencionados anteriormente. Si ha enlazado el software contra libmcrypt-2.2.x, los cuatro comandos importantes de mcrypt (**mcrypt_cfb()**, **mcrypt_cbc()**, **mcrypt_ecb()**, y **mcrypt_ofb()**) pueden operar en ambos modos, los cuales son llamados MCRYPT_ENCRYPT y MCRYPT_DECRYPT, respectivamente.

Ejemplo 1. Cifrar un valor de entrada con TripleDES bajo 2.2.x en modo ECB

```
<?php
$llave = "esta es una llave secreta";
$entrada = "Encontr&eacute;monos a las 9 en punto en el lugar secreto.";

$datos_cifrados = mcrypt_ecb (MCRYPT_3DES, $llave, $entrada, MCRYPT_ENCRYPT);
?>
```

Este ejemplo le entregará los datos cifrados como una cadena en *$datos_cifrados*.

Si ha enlazado el software contra libmcrypt 2.4.x o 2.5.x, éstas funciones aun están disponibles, pero es recomendable que use las funciones avanzadas.

Ejemplo 2. Encriptar un valor de entrada con TripleDES bajo 2.4.x y versiones superiores, en modo ECB

```
<?php
    $llave = "esta es una llave secreta";
    $entrada = "Encontr&eacute;monos a las 9 en punto en el lugar secreto.";

    $td = mcrypt_module_open('tripledes', '', 'ecb', '');
    $iv = mcrypt_create_iv (mcrypt_enc_get_iv_size($td), MCRYPT_RAND);
    mcrypt_generic_init($td, $llave, $iv);
    $datos_cifrados = mcrypt_generic($td, $entrada);
    mcrypt_generic_deinit($td);
    mcrypt_module_close($td);
?>
```

Este ejemplo le entregará los datos cifrados como una cadena en *$datos_cifrados*. Para un ejemplo completo, consulte **mcrypt_module_open()**.

Tabla de contenidos

mcrypt_get_block_size -- Obtiene el tamaño de bloque del cifrado indicado
mcrypt_get_cipher_name -- Obtiene el nombre del cifrado especificado
mcrypt_get_iv_size -- Returns the size of the IV belonging to a specific cipher/mode combination
mcrypt_get_key_size -- Obtiene el tamaño de la clave de un cifrado
mcrypt_list_algorithms -- Get an array of all supported ciphers
mcrypt_list_modes -- Get an array of all supported modes
mcrypt_module_close -- Close the mcrypt module
mcrypt_module_get_algo_block_size -- Returns the blocksize of the specified algorithm
mcrypt_module_get_algo_key_size -- Returns the maximum supported keysize of the opened mode
mcrypt_module_get_supported_key_sizes -- Returns an array with the supported keysizes of the opened algorithm
mcrypt_module_is_block_algorithm_mode -- Returns if the specified module is a block algorithm or not
mcrypt_module_is_block_algorithm -- This function checks whether the specified algorithm is a block algorithm
mcrypt_module_is_block_mode -- Returns if the specified mode outputs blocks or not
mcrypt_module_open -- Opens the module of the algorithm and the mode to be used
mcrypt_module_self_test -- This function runs a self test on the specified module
mcrypt_ofb -- Encripta/desencripta datos en modo OFB
mdecrypt_generic -- Decrypt data

mcrypt_cbc

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

mcrypt_cbc -- Encripta/desencripta datos en modo CBC

Descripción

int **mcrypt_cbc** (int cipher, string key, string data, int mode [, string iv])

mcrypt_cbc() encripta o desencripta (dependiendo de *mode*) los datos *data* con el cifrado *cipher* y la clave *key* en el modo de cifrado CBC y devuelve la cadena resultante.

El parámetro *cipher* es una de las constantes con nombre MCRYPT_nombrecifrado.

key es la clave suministrada al algoritmo. Debe guardarse en secreto.

data son los datos que serán encriptados/desencriptados.

mode es MCRYPT_ENCRYPT o MCRYPT_DECRYPT.

iv es el vector de inicialización opcional.

Ver también: **mcrypt_cfb()**, **mcrypt_ecb()**, **mcrypt_ofb()**

mcrypt_cfb

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

mcrypt_cfb -- Encripta/desencripta datos en modo CFB

Descripción

int **mcrypt_cfb** (int cipher, string key, string data, int mode, string iv)

mcrypt_cfb() encripta o desencripta (dependiendo de *mode*) los datos *data* con el cifrado *cipher* y la clave *key* en el modo de cifrado CFB y devuelve la cadena resultante.

El parámetro *cipher* es una de las constantes con nombre MCRYPT_nombrecifrado.

key es la clave suministrada al algoritmo. Debe guardarse en secreto.

data son los datos que serán encriptados/desencriptados.

mode es MCRYPT_ENCRYPT o MCRYPT_DECRYPT.

iv es el vector de inicialización.

Ver también: **mcrypt_cbc()**, **mcrypt_ecb()**, **mcrypt_ofb()**

mcrypt_create_iv

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

mcrypt_create_iv -- Crea un vector de inicialización (IV) a partir de una fuente aleatoria

Descripción

string **mcrypt_create_iv** (int size, int source)

mcrypt_create_iv() se usa para crear un IV.

mcrypt_create_iv() toma dos argumentos, *size* determina el tamaño del IV, *source* especifica la fuente del IV.

La fuente puede ser MCRYPT_RAND (generador de números aleatorios del sistema), MCRYPT_DEV_RANDOM (que lee datos de /dev/random) y MCRYPT_DEV_URANDOM (que lee datos de /dev/urandom). Si usas MCRYPT_RAND, asegurate de llamar antes a srand() para inicializar el generador de números aleatorios.

Ejemplo 1. Ejemplo de mcrypt_create_iv

```
<?php
$cipher = MCRYPT_TripleDES;
$block_size = mcrypt_get_block_size($cipher);
$iv = mcrypt_create_iv($block_size, MCRYPT_DEV_RANDOM);
?>
```

mcrypt_decrypt

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_decrypt -- Decrypts crypttext with given parameters

Description

string **mcrypt_decrypt** (string cipher, string key, string data, string mode [, string iv])

mcrypt_decrypt() decrypts the data and returns the unencrypted data.

Cipher is one of the MCRYPT_ciphername constants of the name of the algorithm as string.

Key is the key with which the data is encrypted. If it's smaller that the required keysize, it is padded with '*0*'.

Data is the data that will be decrypted with the given cipher and mode. If the size of the data is not n * blocksize, the data will be padded with '*0*'.

Mode is one of the MCRYPT_MODE_modename constants of one of "ecb", "cbc", "cfb", "ofb", "nofb" or "stream".

The *IV* parameter is used for the initialisation in CBC, CFB, OFB modes, and in some algorithms in STREAM mode. If you do not supply an IV, while it is needed for an algorithm, the function issues a warning and uses an IV with all bytes set to '*0*'.

mcrypt_ecb

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

mcrypt_ecb -- Encripta/desencripta datos en modo ECB

Descripción

int **mcrypt_ecb** (int cipher, string key, string data, int mode)

mcrypt_ecb() encripta o desencripta (dependiendo de *mode*) los datos *data* con el cifrado *cipher* y la clave *key* en el modo de cifrado ECB y devuelve la cadena resultante.

El parámetro *cipher* es una de las constantes con nombre MCRYPT_nombrecifrado.

key es la clave suministrada al algoritmo. Debe guardarse en secreto.

data son los datos que serán encriptados/desencriptados.

mode es MCRYPT_ENCRYPT o MCRYPT_DECRYPT.

Ver también: **mcrypt_cbc()**, **mcrypt_cfb()**, **mcrypt_ofb()**

mcrypt_enc_get_algorithms_name

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_enc_get_algorithms_name -- Returns the name of the opened algorithm

Description

string **mcrypt_enc_get_algorithms_name** (resource td)

This function returns the name of the algorithm.

Ejemplo 1. mcrypt_enc_get_algorithms_name() example

```
<?php
$td = mcrypt_module_open(MCRYPT_CAST_256, '', MCRYPT_MODE_CFB, '');
echo mcrypt_enc_get_algorithms_name($td). "\n";

$td = mcrypt_module_open('cast-256', '', MCRYPT_MODE_CFB, '');
echo mcrypt_enc_get_algorithms_name($td). "\n";
?>
```

El resultado del ejemplo seria:

```
CAST-256
CAST-256
```

mcrypt_enc_get_block_size

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_enc_get_block_size -- Returns the blocksize of the opened algorithm

Description

int **mcrypt_enc_get_block_size** (resource td)

This function returns the block size of the algorithm specified by the encryption descriptor *td* in bytes.

mcrypt_enc_get_iv_size

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_enc_get_iv_size -- Returns the size of the IV of the opened algorithm

Description

int **mcrypt_enc_get_iv_size** (resource td)

This function returns the size of the iv of the algorithm specified by the encryption descriptor in bytes. If it returns '0' then the IV is ignored in the algorithm. An IV is used in cbc, cfb and ofb modes, and in some algorithms in stream mode.

mcrypt_enc_get_key_size

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_enc_get_key_size -- Returns the maximum supported keysize of the opened mode

Description

int **mcrypt_enc_get_key_size** (resource td)

This function returns the maximum supported key size of the algorithm specified by the encryption descriptor *td* in bytes.

mcrypt_enc_get_modes_name

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_enc_get_modes_name -- Returns the name of the opened mode

Description

string **mcrypt_enc_get_modes_name** (resource td)

This function returns the name of the mode.

Ejemplo 1. mcrypt_enc_get_modes_name() example

```
<?php
    $td = mcrypt_module_open (MCRYPT_CAST_256, '', MCRYPT_MODE_CFB, '');
    echo mcrypt_enc_get_modes_name($td). "\n";

    $td = mcrypt_module_open ('cast-256', '', 'ecb', '');
    echo mcrypt_enc_get_modes_name($td). "\n";
?>
```

Prints:

```
CFB
ECB
```

mcrypt_enc_get_supported_key_sizes

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_enc_get_supported_key_sizes -- Returns an array with the supported keysizes of the opened algorithm

Description

array **mcrypt_enc_get_supported_key_sizes** (resource td)

Returns an array with the key sizes supported by the algorithm specified by the encryption descriptor. If it returns an empty array then all key sizes between 1 and **mcrypt_enc_get_key_size()** are supported by the algorithm.

Ejemplo 1. mcrypt_enc_get_supported_key_sizes() example

```
<?php
    $td = mcrypt_module_open('rijndael-256', '', 'ecb', '');
    var_dump(mcrypt_enc_get_supported_key_sizes($td));
?>
```

This will print:

```
array(3) {
  [0]=>
  int(16)
  [1]=>
  int(24)
  [2]=>
  int(32)
}
```

mcrypt_enc_is_block_algorithm_mode

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_enc_is_block_algorithm_mode -- Checks whether the encryption of the opened mode works on blocks

Description

bool **mcrypt_enc_is_block_algorithm_mode** (resource td)

This function returns **TRUE** if the mode is for use with block algorithms, otherwise it returns **FALSE**. (e.g. **FALSE** for stream, and **TRUE** for cbc, cfb, ofb).

mcrypt_enc_is_block_algorithm

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_enc_is_block_algorithm -- Checks whether the algorithm of the opened mode is a block algorithm

Description

bool **mcrypt_enc_is_block_algorithm** (resource td)

This function returns **TRUE** if the algorithm is a block algorithm, or **FALSE** if it is a stream algorithm.

mcrypt_enc_is_block_mode

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_enc_is_block_mode -- Checks whether the opened mode outputs blocks

Description

bool **mcrypt_enc_is_block_mode** (resource td)

This function returns **TRUE** if the mode outputs blocks of bytes or **FALSE** if it outputs bytes. (e.g. **TRUE** for cbc and ecb, and **FALSE** for cfb and stream).

mcrypt_enc_self_test

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_enc_self_test -- This function runs a self test on the opened module

Description

bool **mcrypt_enc_self_test** (resource td)

This function runs the self test on the algorithm specified by the descriptor *td*. If the self test succeeds it returns **FALSE**. In case of an error, it returns **TRUE**.

mcrypt_encrypt

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_encrypt -- Encrypts plaintext with given parameters

Description

string **mcrypt_encrypt** (string cipher, string key, string data, string mode [, string iv])

mcrypt_encrypt() encrypts the data and returns the encrypted data.

Cipher is one of the MCRYPT_ciphername constants of the name of the algorithm as string.

Key is the key with which the data will be encrypted. If it's smaller that the required keysize, it is padded with '\0'. It is better not to use ASCII strings for keys. It is recommended to use the mhash functions to create a key from a string.

Data is the data that will be encrypted with the given cipher and mode. If the size of the data is not n * blocksize, the data will be padded with '\0'. The returned crypttext can be larger that the size of the data that is given by *data*.

Mode is one of the MCRYPT_MODE_modename constants of one of "ecb", "cbc", "cfb", "ofb", "nofb" or "stream".

The *IV* parameter is used for the initialisation in CBC, CFB, OFB modes, and in some algorithms in STREAM mode. If you do not supply an IV, while it is needed for an algorithm, the function issues a warning and uses an IV with all bytes set to '\0'.

Ejemplo 1. mcrypt_encrypt() Example

```
<?php
    $iv_size = mcrypt_get_iv_size(MCRYPT_RIJNDAEL_256, MCRYPT_MODE_ECB);
    $iv = mcrypt_create_iv($iv_size, MCRYPT_RAND);
    $key = "This is a very secret key";
    $text = "Meet me at 11 o'clock behind the monument.";
    echo strlen($text) . "\n";

    $crypttext = mcrypt_encrypt(MCRYPT_RIJNDAEL_256, $key, $text, MCRYPT_MODE_ECB, $iv);
    echo strlen($crypttext) . "\n";
?>
```

The above example will print out:

```
42
64
```

See also **mcrypt_module_open()** for a more advanced API and an example.

mcrypt_generic_deinit

(PHP 4 >= 4.1.1, PHP 5)

mcrypt_generic_deinit -- This function deinitializes an encryption module

Description

bool **mcrypt_generic_deinit** (resource td)

This function terminates encryption specified by the encryption descriptor (*td*). It clears all buffers, but does not close the module. You need to call **mcrypt_module_close()** yourself. (But PHP does this for you at the end of the script.) Returns **FALSE** on error, or **TRUE** on success.

See for an example **mcrypt_module_open()** and the entry on **mcrypt_generic_init()**.

mcrypt_generic_end

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_generic_end -- This function terminates encryption

Description

bool **mcrypt_generic_end** (resource td)

Aviso
This function is deprecated, use **mcrypt_generic_deinit()** instead. It can cause crashes when used with **mcrypt_module_close()** due to multiple buffer frees.

This function terminates encryption specified by the encryption descriptor (*td*). Actually it clears all buffers, and closes all the modules used. Returns **FALSE** on error, or **TRUE** on success.

mcrypt_generic_init

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_generic_init -- This function initializes all buffers needed for encryption

Description

int **mcrypt_generic_init** (resource td, string key, string iv)

The maximum length of the key should be the one obtained by calling **mcrypt_enc_get_key_size()**

and every value smaller than this is legal. The IV should normally have the size of the algorithms block size, but you must obtain the size by calling **mcrypt_enc_get_iv_size()**. IV is ignored in ECB. IV MUST exist in CFB, CBC, STREAM, nOFB and OFB modes. It needs to be random and unique (but not secret). The same IV must be used for encryption/decryption. If you do not want to use it you should set it to zeros, but this is not recommended.

The function returns a negative value on error, -3 when the key length was incorrect, -4 when there was a memory allocation problem and any other return value is an unknown error. If an error occurs a warning will be displayed accordingly. **FALSE** is returned if incorrect parameters were passed.

You need to call this function before every call to **mcrypt_generic()** or **mdecrypt_generic()**.

See for an example **mcrypt_module_open()**.

mcrypt_generic

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_generic -- This function encrypts data

Description

string **mcrypt_generic** (resource td, string data)

This function encrypts data. The data is padded with "\0" to make sure the length of the data is n * blocksize. This function returns the encrypted data. Note that the length of the returned string can in fact be longer then the input, due to the padding of the data.

If you want to store the encrypted data in a database make sure to store the entire string as returned by mcrypt_generic, or the string will not entirely decrypt properly. If your original string is 10 characters long and the block size is 8 (use **mcrypt_enc_get_block_size()** to determine the blocksize), you would need at least 16 characters in your database field. Note the string returned by **mdecrypt_generic()** will be 16 characters as well...use **rtrim()**($str, "\0") to remove the padding.

If you are for example storing the data in a MySQL database remember that varchar fields automatically have trailing spaces removed during insertion. As encrypted data can end in a space (ASCII 32), the data will be damaged by this removal. Store data in a tinyblob/tinytext (or larger) field instead.

The encryption handle should always be initialized with **mcrypt_generic_init()** with a key and an IV before calling this function. Where the encryption is done, you should free the encryption buffers by calling **mcrypt_generic_deinit()**. See **mcrypt_module_open()** for an example.

See also **mdecrypt_generic()**, **mcrypt_generic_init()**, and **mcrypt_generic_deinit()**.

mcrypt_get_block_size

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

mcrypt_get_block_size -- Obtiene el tamaño de bloque del cifrado indicado

Descripción

int **mcrypt_get_block_size** (int cipher)

mcrypt_get_block_size() se usa para obtener el tamaño de bloque del cifrado indicado en *cipher*.

mcrypt_get_block_size() toma un argumento, el cifrado *cipher* y devuelve el tamaño en bytes.

Ver también: **mcrypt_get_key_size()**

mcrypt_get_cipher_name

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

mcrypt_get_cipher_name -- Obtiene el nombre del cifrado especificado

Descripción

string **mcrypt_get_cipher_name** (int cipher)

mcrypt_get_cipher_name() se usa para obtener el nombre del cifrado especificado.

mcrypt_get_cipher_name() toma como argumento el número de cifrado y devuelve el nombre del cifrado o **FALSE**, si el cifrado no existe.

Ejemplo 1. Ejemplo de mcrypt_get_cipher_name

```
<?php
$cipher = MCRYPT_TripleDES;

print mcrypt_get_cipher_name($cipher);
?>
```

El ejemplo de más arriba da como resultado:

```
TripleDES
```

mcrypt_get_iv_size

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_get_iv_size -- Returns the size of the IV belonging to a specific cipher/mode combination

Description

int **mcrypt_get_iv_size** (string cipher, string mode)

mcrypt_get_iv_size() returns the size of the Initialisation Vector (IV) in bytes. On error the function returns **FALSE**. If the IV is ignored in the specified cipher/mode combination zero is returned.

cipher is one of the MCRYPT_ciphername constants of the name of the algorithm as string.

mode is one of the MCRYPT_MODE_modename constants or one of "ecb", "cbc", "cfb", "ofb",

"nofb" or "stream". The IV is ignored in ECB mode as this mode does not require it. You will need to have the same IV (think: starting point) both at encryption and decryption stages, otherwise your encryption will fail.

It is more useful to use the **mcrypt_enc_get_iv_size()** function as this uses the resource returned by **mcrypt_module_open()**.

Ejemplo 1. mcrypt_get_iv_size() example

```
<?php
    echo mcrypt_get_iv_size(MCRYPT_CAST_256, MCRYPT_MODE_CFB) . "\n";

    echo mcrypt_get_iv_size('des', 'ecb') . "\n";
?>
```

See also **mcrypt_get_block_size()**, **mcrypt_enc_get_iv_size()** and **mcrypt_create_iv()**.

mcrypt_get_key_size

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

mcrypt_get_key_size -- Obtiene el tamaño de la clave de un cifrado

Descripción

int **mcrypt_get_key_size** (int cipher)

mcrypt_get_key_size() se usa para obtener el tamaño de la clave del cifrado indicado en *cipher*.

mcrypt_get_key_size() toma un argumento, el cifrado *cipher* y devuelve el tamaño de la clave en bytes.

Ver también: **mcrypt_get_block_size()**

mcrypt_list_algorithms

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_list_algorithms -- Get an array of all supported ciphers

Description

array **mcrypt_list_algorithms** ([string lib_dir])

mcrypt_list_algorithms() is used to get an array of all supported algorithms in the *lib_dir* parameter.

mcrypt_list_algorithms() takes an optional *lib_dir* parameter which specifies the directory where all algorithms are located. If not specifies, the value of the mcrypt.algorithms_dir `php.ini` directive is used.

Ejemplo 1. mcrypt_list_algorithms() Example

```
<?php
    $algorithms = mcrypt_list_algorithms("/usr/local/lib/libmcrypt");

    foreach ($algorithms as $cipher) {
        echo "$cipher<br />\n";
    }
?>
```

The above example will produce a list with all supported algorithms in the "/usr/local/lib/libmcrypt" directory.

mcrypt_list_modes

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_list_modes -- Get an array of all supported modes

Description

array **mcrypt_list_modes** ([string lib_dir])

mcrypt_list_modes() is used to get an array of all supported modes in the *lib_dir*.

mcrypt_list_modes() takes as optional parameter a directory which specifies the directory where all modes are located. If not specifies, the value of the mcrypt.modes_dir `php.ini` directive is used.

Ejemplo 1. mcrypt_list_modes() Example

```
<?php
    $modes = mcrypt_list_modes();

    foreach ($modes as $mode) {
        echo "$mode <br />\n";
    }
?>
```

The above example will produce a list with all supported algorithms in the default mode directory. If it is not set with the ini directive mcrypt.modes_dir, the default directory of mcrypt is used (which is /usr/local/lib/libmcrypt).

mcrypt_module_close

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_module_close -- Close the mcrypt module

Description

bool **mcrypt_module_close** (resource td)

This function closes the specified encryption handle.

See **mcrypt_module_open()** for an example.

mcrypt_module_get_algo_block_size

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_module_get_algo_block_size -- Returns the blocksize of the specified algorithm

Description

int **mcrypt_module_get_algo_block_size** (string algorithm [, string lib_dir])

This function returns the block size of the algorithm specified in bytes. The optional *lib_dir* parameter can contain the location where the mode module is on the system.

mcrypt_module_get_algo_key_size

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_module_get_algo_key_size -- Returns the maximum supported keysize of the opened mode

Description

int **mcrypt_module_get_algo_key_size** (string algorithm [, string lib_dir])

This function returns the maximum supported key size of the algorithm specified in bytes. The optional *lib_dir* parameter can contain the location where the mode module is on the system.

mcrypt_module_get_supported_key_sizes

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_module_get_supported_key_sizes -- Returns an array with the supported keysizes of the opened algorithm

Description

array **mcrypt_module_get_supported_key_sizes** (string algorithm [, string lib_dir])

Returns an array with the key sizes supported by the specified algorithm. If it returns an empty array then all key sizes between 1 and **mcrypt_module_get_algo_key_size()** are supported by the algorithm. The optional *lib_dir* parameter can contain the location where the mode module is on the system.

See also **mcrypt_enc_get_supported_key_sizes()** which is used on open encryption modules.

mcrypt_module_is_block_algorithm_mode

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_module_is_block_algorithm_mode -- Returns if the specified module is a block algorithm

or not

Description

bool **mcrypt_module_is_block_algorithm_mode** (string mode [, string lib_dir])

This function returns **TRUE** if the mode is for use with block algorithms, otherwise it returns **FALSE**. (e.g. **FALSE** for stream, and **TRUE** for cbc, cfb, ofb). The optional *lib_dir* parameter can contain the location where the mode module is on the system.

mcrypt_module_is_block_algorithm

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_module_is_block_algorithm -- This function checks whether the specified algorithm is a block algorithm

Description

bool **mcrypt_module_is_block_algorithm** (string algorithm [, string lib_dir])

This function returns **TRUE** if the specified algorithm is a block algorithm, or **FALSE** is it is a stream algorithm. The optional *lib_dir* parameter can contain the location where the algorithm module is on the system.

mcrypt_module_is_block_mode

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_module_is_block_mode -- Returns if the specified mode outputs blocks or not

Description

bool **mcrypt_module_is_block_mode** (string mode [, string lib_dir])

This function returns **TRUE** if the mode outputs blocks of bytes or **FALSE** if it outputs just bytes. (e.g. **TRUE** for cbc and ecb, and **FALSE** for cfb and stream). The optional *lib_dir* parameter can contain the location where the mode module is on the system.

mcrypt_module_open

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_module_open -- Opens the module of the algorithm and the mode to be used

Description

resource **mcrypt_module_open** (string algorithm, string algorithm_directory, string mode, string mode_directory)

This function opens the module of the algorithm and the mode to be used. The name of the algorithm is specified in algorithm, e.g. "twofish" or is one of the MCRYPT_ciphername constants. The module is closed by calling **mcrypt_module_close()**. Normally it returns an encryption descriptor, or **FALSE** on error.

The *algorithm_directory* and *mode_directory* are used to locate the encryption modules. When you supply a directory name, it is used. When you set one of these to the empty string (""), the value set by the *mcrypt.algorithms_dir* or *mcrypt.modes_dir* ini-directive is used. When these are not set, the default directories that are used are the ones that were compiled in into libmcrypt (usually / usr/local/lib/libmcrypt).

Ejemplo 1. mcrypt_module_open() examples

```
<?php
    $td = mcrypt_module_open(MCRYPT_DES, '',
        MCRYPT_MODE_ECB, '/usr/lib/mcrypt-modes');

    $td = mcrypt_module_open('rijndael-256', '', 'ofb', '');
?>
```

The first line in the example above will try to open the DES cipher from the default directory and the EBC mode from the directory `/usr/lib/mcrypt-modes`. The second example uses strings as name for the cipher and mode, this only works when the extension is linked against libmcrypt 2.4.x or 2.5.x.

Ejemplo 2. Using mcrypt_module_open() in encryption

```
<?php
    /* Open the cipher */
    $td = mcrypt_module_open('rijndael-256', '', 'ofb', '');

    /* Create the IV and determine the keysize length, used MCRYPT_RAND
     * on Windows instead */
    $iv = mcrypt_create_iv(mcrypt_enc_get_iv_size($td), MCRYPT_DEV_RANDOM);
    $ks = mcrypt_enc_get_key_size($td);

    /* Create key */
    $key = substr(md5('very secret key'), 0, $ks);

    /* Intialize encryption */
    mcrypt_generic_init($td, $key, $iv);

    /* Encrypt data */
    $encrypted = mcrypt_generic($td, 'This is very important data');

    /* Terminate encryption handler */
    mcrypt_generic_deinit($td);

    /* Initialize encryption module for decryption */
    mcrypt_generic_init($td, $key, $iv);

    /* Decrypt encrypted string */
    $decrypted = mdecrypt_generic($td, $encrypted);

    /* Terminate decryption handle and close module */
    mcrypt_generic_deinit($td);
    mcrypt_module_close($td);

    /* Show string */
    echo trim($decrypted) . "\n";
?>
```

The first line in the example above will try to open the DES cipher from the default directory and the EBC mode from the directory `/usr/lib/mcrypt-modes`. The second example uses strings as name for the cipher and mode, this only works when the extension is linked against libmcrypt 2.4.x or 2.5.x.

See also **mcrypt_module_close()**, **mcrypt_generic()**, **mdecrypt_generic()**, **mcrypt_generic_init()**, and **mcrypt_generic_deinit()**.

mcrypt_module_self_test

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_module_self_test -- This function runs a self test on the specified module

Description

bool **mcrypt_module_self_test** (string algorithm [, string lib_dir])

This function runs the self test on the algorithm specified. The optional *lib_dir* parameter can contain the location of where the algorithm module is on the system.

The function returns **TRUE** if the self test succeeds, or **FALSE** when if fails.

Ejemplo 1. mcrypt_module_self_test() example

```
<?php
var_dump(mcrypt_module_self_test(MCRYPT_RIJNDAEL_128)) . "\n";
var_dump(mcrypt_module_self_test(MCRYPT_BOGUS_CYPHER));
?>
```

The above example will output:

```
bool(true)
bool(false)
```

mcrypt_ofb

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

mcrypt_ofb -- Encripta/desencripta datos en modo OFB

Descripción

int **mcrypt_ofb** (int cipher, string key, string data, int mode, string iv)

mcrypt_ofb() encripta o desencripta (dependiendo de *mode*) los datos *data* con el cifrado *cipher* y la clave *key* en el modo de cifrado OFB y devuelve la cadena resultante.

El parámetro *cipher* es una de las constantes con nombre MCRYPT_nombrecifrado.

key es la clave suministrada al algoritmo. Debe guardarse en secreto.

data son los datos que serán encriptados/desencriptados.

mode es MCRYPT_ENCRYPT o MCRYPT_DECRYPT.

iv es el vector de inicialización.

Ver también: **mcrypt_cbc()**, **mcrypt_cfb()**, **mcrypt_ecb()**

mdecrypt_generic

(PHP 4 >= 4.0.2, PHP 5)

mdecrypt_generic -- Decrypt data

Description

string **mdecrypt_generic** (resource td, string data)

This function decrypts data. Note that the length of the returned string can in fact be longer then the unencrypted string, due to the padding of the data.

Ejemplo 1. mdecrypt_generic() example

```
<?php
    /* Data */
    $key = 'this is a very long key, even too long for the cipher';
    $plain_text = 'very important data';

    /* Open module, and create IV */
    $td = mcrypt_module_open('des', '', 'ecb', '');
    $key = substr($key, 0, mcrypt_enc_get_key_size($td));
    $iv_size = mcrypt_enc_get_iv_size($td);
    $iv = mcrypt_create_iv($iv_size, MCRYPT_RAND);

    /* Initialize encryption handle */
    if (mcrypt_generic_init($td, $key, $iv) != -1) {

        /* Encrypt data */
        $c_t = mcrypt_generic($td, $plain_text);
        mcrypt_generic_deinit($td);

        /* Reinitialize buffers for decryption */
        mcrypt_generic_init($td, $key, $iv);
        $p_t = mdecrypt_generic($td, $c_t);

        /* Clean up */
        mcrypt_generic_deinit($td);
        mcrypt_module_close($td);
    }

    if (strncmp($p_t, $plain_text, strlen($plain_text)) == 0) {
        echo "ok\n";
    } else {
        echo "error\n";
    }
?>
```

The above example shows how to check if the data before the encryption is the same as the data after the decryption. It is very important to reinitialize the encryption buffer with **mcrypt_generic_init()** before you try to decrypt the data.

The decryption handle should always be initialized with **mcrypt_generic_init()** with a key and an IV before calling this function. Where the encryption is done, you should free the encryption buffers by calling **mcrypt_generic_deinit()**. See **mcrypt_module_open()** for an example.

See also **mcrypt_generic()**, **mcrypt_generic_init()**, and **mcrypt_generic_deinit()**.

LXVIII. MCVE Payment Functions

Introducción

These functions interface the MCVE API (libmcve), allowing you to work directly with MCVE from your PHP scripts. MCVE is Main Street Softworks' solution to direct credit card processing for Linux / Unix (http://www.mainstreetsoftworks.com/). It lets you directly address the credit card clearing houses via your *nix box, modem and/or internet connection (bypassing the need for an additional service such as Authorize.Net or Pay Flow Pro). Using the MCVE module for PHP, you can process credit cards directly through MCVE via your PHP scripts. The following references will outline the process.

> **Nota:** MCVE is the replacement for RedHat's CCVS. They contracted with RedHat in late 2001 to migrate all existing clientele to the MCVE platform.

> **Nota:** Esta extensión no está disponible en plataformas Windows

---

Instalación

To enable MCVE Support in PHP, first verify your LibMCVE installation directory. You will then need to configure PHP with the *--with-mcve* option. If you use this option without specifying the path to your MCVE installation, PHP will attempt to look in the default LibMCVE Install location (`/usr/local`). If MCVE is in a non-standard location, run configure with: *--with-mcve=$mcve_path*, where $mcve_path is the path to your MCVE installation. Please note that MCVE support requires that $mcve_path/lib and $mcve_path/include exist, and include `mcve.h` under the include directory and `libmcve.so` and/or `libmcve.a` under the lib directory.

Since MCVE has true server/client separation, there are no additional requirements for running PHP with MCVE support. To test your MCVE extension in PHP, you may connect to testbox.mcve.com on port 8333 for IP, or port 8444 for SSL using the MCVE PHP API. Use 'vitale' for your username, and 'test' for your password. Additional information about test facilities are available at http://www.mainstreetsoftworks.com/.

Ver también

Additional documentation about MCVE's PHP API can be found at http://www.mainstreetsoftworks.com/docs/phpapi.pdf. Main Street's documentation is complete and should be the primary reference for functions.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

`MC_TRANTYPE` (**integer**)

MC_USERNAME (integer)

MC_PASSWORD (integer)

MC_ACCOUNT (integer)

MC_TRACKDATA (integer)

MC_EXPDATE (integer)

MC_STREET (integer)

MC_ZIP (integer)

MC_CV (integer)

MC_COMMENTS (integer)

MC_CLERKID (integer)

MC_STATIONID (integer)

MC_APPRCODE (integer)

MC_AMOUNT (integer)

MC_PTRANNUM (integer)

MC_TTID (integer)

MC_USER (integer)

MC_PWD (integer)

MC_ACCT (integer)

MC_BDATE (integer)

MC_EDATE (integer)

MC_BATCH (integer)

MC_FILE (integer)

MC_ADMIN (integer)

MC_AUDITTYPE (integer)

MC_CUSTOM (integer)

MC_EXAMOUNT (integer)

MC_EXCHARGES (integer)

MC_RATE (integer)

MC_RENTERNAME (integer)

MC_RETURNCITY (integer)

MC_RETURNSTATE (integer)

MC_RETURNLOCATION (integer)

MC_PRIORITY (integer)

MC_INQUIRY (integer)

MC_CARDTYPES (integer)

MC_SUB (integer)

MC_MARKER (integer)

MC_DEVICETYPE (integer)

MC_ERRORCODE (integer)

MC_NEWBATCH (integer)

MC_CURR (integer)

MC_DESCMERCH (integer)

MC_DESCLOC (integer)

MC_ORIGTYPE (integer)

MC_PIN (integer)

MC_VOIDORIGTYPE (integer)

MC_TIMESTAMP (integer)

MC_PRIO_HIGH (integer)

MC_PRIO_NORMAL (integer)

MC_PRIO_LOW (integer)

MC_USER_PROC (integer)

MC_USER_USER (integer)

MC_USER_PWD (integer)

MC_USER_INDCODE (integer)

MC_USER_MERCHID (integer)

MC_USER_BANKID (integer)

MC_USER_TERMID (integer)

MC_USER_CLIENTNUM (integer)

MC_USER_STOREID (integer)

MC_USER_AGENTID (integer)

MC_USER_CHAINID (integer)

MC_USER_ZIPCODE (integer)

MC_USER_TIMEZONE (integer)

MC_USER_MERCHCAT (integer)

MC_USER_MERNAME (integer)

MC_USER_MERCHLOC (integer)

MC_USER_STATECODE (integer)

MC_USER_PHONE (integer)

MC_USER_SUB (integer)

MC_USER_CARDTYPES (integer)

MC_USER_MODE (integer)

MC_USER_VNUMBER (integer)

MC_USER_ROUTINGID (integer)

MC_USER_PPROPERTY (integer)

MC_USER_PID (integer)

MC_USER_PIDPWD (integer)

MC_USER_SMID (integer)

MC_USER_SMIDPWD (integer)

MC_USER_USDDIV (integer)

MC_USER_AUDDIV (integer)

MC_USER_DKKDIV (integer)

MC_USER_GBPDIV (integer)

MC_USER_HKDDIV (integer)

MC_USER_JPYDIV (integer)

MC_USER_NZDDIV (integer)

MC_USER_NOKDIV (integer)

MC_USER_SGDDIV (integer)

MC_USER_ZARDIV (integer)

MC_USER_SEKDIV (integer)

MC_USER_CHFDIV (integer)

MC_USER_CADDIV (integer)

MC_USER_DIVNUM (integer)

MC_CARD_VISA (integer)

MC_CARD_MC (integer)

MC_CARD_AMEX (integer)

MC_CARD_DISC (integer)

MC_CARD_JCB (integer)

MC_CARD_CB (integer)

MC_CARD_DC (integer)

MC_CARD_GIFT (integer)

MC_CARD_OTHER (integer)

MC_CARD_ALL (integer)

MC_MODE_AUTH (integer)

MC_MODE_SETTLE (integer)

MC_MODE_BOTH (integer)

MC_MODE_ALL (integer)

MC_EXCHARGES_REST (integer)

MC_EXCHARGES_GIFT (integer)

MC_EXCHARGES_MINI (integer)

MC_EXCHARGES_TELE (integer)

MC_EXCHARGES_OTHER (integer)

MC_EXCHARGES_LAUND (integer)

MC_EXCHARGES_NONE (integer)

MC_EXCHARGES_GAS (integer)

MC_EXCHARGES_MILE (integer)

MC_EXCHARGES_LATE (integer)

MC_EXCHARGES_1WAY (integer)

MC_EXCHARGES_VIOL (integer)

MC_TRAN_SALE (integer)

MC_TRAN_REDEMPTION (integer)

MC_TRAN_PREAUTH (integer)

MC_TRAN_VOID (integer)

MC_TRAN_PREAUTHCOMPLETE (integer)

MC_TRAN_FORCE (integer)

MC_TRAN_OVERRIDE (integer)

MC_TRAN_RETURN (integer)

MC_TRAN_RELOAD (integer)

MC_TRAN_CREDIT (integer)

MC_TRAN_SETTLE (integer)

MC_TRAN_INCREMENTAL (integer)

MC_TRAN_REVERSAL (integer)

MC_TRAN_ACTIVATE (integer)

MC_TRAN_BALANCEINQ (integer)

MC_TRAN_CASHOUT (integer)

MC_TRAN_TOREVERSAL (integer)

MC_TRAN_SETTLERFR (integer)

MC_TRAN_ISSUE (integer)

MC_TRAN_TIP (integer)

MC_TRAN_MERCHRETURN (integer)

MC_TRAN_IVRREQ (integer)

MC_TRAN_IVRRESP (integer)

MC_TRAN_ADMIN (integer)

MC_TRAN_PING (integer)

MC_TRAN_CHKPWD (integer)

MC_TRAN_CHNGPWD (integer)

MC_TRAN_LISTSTATS (integer)

MC_TRAN_LISTUSERS (integer)

MC_TRAN_GETUSERINFO (integer)

MC_TRAN_ADDUSER (integer)

MC_TRAN_EDITUSER (integer)

MC_TRAN_DELUSER (integer)

MC_TRAN_ENABLEUSER (integer)

MC_TRAN_DISABLEUSER (integer)

MC_TRAN_IMPORT (integer)

MC_TRAN_EXPORT (integer)

MC_TRAN_ERRORLOG (integer)

MC_TRAN_CLEARERRORLOG (integer)

MC_TRAN_GETSUBACCTS (integer)

MC_ADMIN_GUT (integer)

MC_ADMIN_GL (integer)

MC_ADMIN_GFT (integer)

MC_ADMIN_BT (integer)

MC_ADMIN_UB (integer)

MC_ADMIN_QC (integer)

MC_ADMIN_RS (integer)

MC_ADMIN_CTH (integer)

MC_ADMIN_CFH (integer)

MC_ADMIN_FORCESETTLE (integer)

MC_ADMIN_SETBATCHNUM (integer)

MC_ADMIN_RENUMBERBATCH (integer)

MC_ADMIN_FIELDEDIT (integer)

MC_ADMIN_CLOSEBATCH (integer)

MCVE_UNUSED (integer)

MCVE_NEW (integer)

MCVE_PENDING (integer)

MCVE_DONE (integer)

MCVE_GOOD (integer)

MCVE_BAD (integer)

MCVE_STREET (integer)

MCVE_ZIP (integer)

MCVE_UNKNOWN (integer)

MCVE_ERROR (integer)

MCVE_FAIL (integer)

MCVE_SUCCESS (integer)

MCVE_AUTH (integer)

MCVE_DENY (integer)

MCVE_CALL (integer)

MCVE_DUPL (integer)

MCVE_PKUP (integer)

MCVE_RETRY (integer)

MCVE_SETUP (integer)

MCVE_TIMEOUT (integer)

MCVE_SALE (integer)

MCVE_PREAUTH (integer)

MCVE_FORCE (integer)

MCVE_OVERRIDE (integer)

MCVE_RETURN (integer)

MCVE_SETTLE (integer)

MCVE_PROC (integer)

MCVE_USER (integer)

MCVE_PWD (integer)

MCVE_INDCODE (integer)

MCVE_MERCHID (integer)

MCVE_BANKID (integer)

MCVE_TERMID (integer)

MCVE_CLIENTNUM (integer)

MCVE_STOREID (integer)

MCVE_AGENTID (integer)

MCVE_CHAINID (**integer**)

MCVE_ZIPCODE (**integer**)

MCVE_TIMEZONE (**integer**)

MCVE_MERCHCAT (**integer**)

MCVE_MERNAME (**integer**)

MCVE_MERCHLOC (**integer**)

MCVE_STATECODE (**integer**)

MCVE_SERVICEPHONE (**integer**)

Tabla de contenidos

mcve_monitor -- Perform communication with MCVE (send/receive data) Non-blocking
mcve_numcolumns -- Number of columns returned in a comma delimited response
mcve_numrows -- Number of rows returned in a comma delimited response
mcve_override -- Send an OVERRIDE to MCVE
mcve_parsecommadelimited -- Parse the comma delimited response so mcve_getcell, etc will work
mcve_ping -- Send a ping request to MCVE
mcve_preauth -- Send a PREAUTHORIZATION to MCVE
mcve_preauthcompletion -- Complete a PREAUTHORIZATION, ready it for settlement
mcve_qc -- Audit MCVE for a list of transactions in the outgoing queue
mcve_responseparam -- Get a custom response parameter
mcve_return -- Issue a RETURN or CREDIT to MCVE
mcve_returncode -- Grab the exact return code from the transaction
mcve_returnstatus -- Check to see if the transaction was successful
mcve_sale -- Send a SALE to MCVE
mcve_setblocking -- Set blocking/non-blocking mode for connection
mcve_setdropfile -- Set the connection method to Drop-File
mcve_setip -- Set the connection method to IP
mcve_setssl_files -- Set certificate key files and certificates if server requires client certificate verification
mcve_setssl -- Set the connection method to SSL
mcve_settimeout -- Set maximum transaction time (per trans)
mcve_settle -- Issue a settlement command to do a batch deposit
mcve_text_avs -- Get a textual representation of the return_avs
mcve_text_code -- Get a textual representation of the return_code
mcve_text_cv -- Get a textual representation of the return_cv
mcve_transactionauth -- Get the authorization number returned for the transaction (alpha-numeric)
mcve_transactionavs -- Get the Address Verification return status
mcve_transactionbatch -- Get the batch number associated with the transaction
mcve_transactioncv -- Get the CVC2/CVV2/CID return status
mcve_transactionid -- Get the unique system id for the transaction
mcve_transactionitem -- Get the ITEM number in the associated batch for this transaction
mcve_transactionssent -- Check to see if outgoing buffer is clear
mcve_transactiontext -- Get verbiage (text) return from MCVE or processing institution
mcve_transinqueue -- Number of transactions in client-queue
mcve_transnew -- Start a new transaction
mcve_transparam -- Add a parameter to a transaction
mcve_transsend -- Finalize and send the transaction
mcve_ub -- Get a list of all Unsettled batches
mcve_uwait -- Wait x microsecs
mcve_verifyconnection -- Set whether or not to PING upon connect to verify connection
mcve_verifysslcert -- Set whether or not to verify the server ssl certificate
mcve_void -- VOID a transaction in the settlement queue

mcve_adduser

(PHP 4 >= 4.2.0, PHP 5)

mcve_adduser -- Add an MCVE user using usersetup structure

Description

int **mcve_adduser** (resource conn, string admin_password, int usersetup)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_adduserarg

(PHP 4 >= 4.2.0, PHP 5)

mcve_adduserarg -- Add a value to user configuration structure

Description

int **mcve_adduserarg** (resource usersetup, int argtype, string argval)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_bt

(PHP 4 >= 4.2.0, PHP 5)

mcve_bt -- Get unsettled batch totals

Description

int **mcve_bt** (resource conn, string username, string password)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_checkstatus

(PHP 4 >= 4.2.0, PHP 5)

mcve_checkstatus -- Check to see if a transaction has completed

Description

int **mcve_checkstatus** (resource conn, int identifier)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_chkpwd

(PHP 4 >= 4.2.0, PHP 5)

mcve_chkpwd -- Verify Password

Description

int **mcve_chkpwd** (resource conn, string username, string password)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_chngpwd

(PHP 4 >= 4.2.0, PHP 5)

mcve_chngpwd -- Change the system administrator's password

Description

int **mcve_chngpwd** (resource conn, string admin_password, string new_password)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_completeauthorizations

(PHP 4 >= 4.2.0, PHP 5)

mcve_completeauthorizations -- Number of complete authorizations in queue, returning an array of their identifiers

Description

int **mcve_completeauthorizations** (resource conn, int &array)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_connect

(PHP 4 >= 4.2.0, PHP 5)

mcve_connect -- Establish the connection to MCVE

Description

int **mcve_connect** (resource conn)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_connectionerror

(PHP 4 >= 4.3.0, PHP 5)

mcve_connectionerror -- Get a textual representation of why a connection failed

Description

string **mcve_connectionerror** (resource conn)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_deleteresponse

(PHP 4 >= 4.2.0, PHP 5)

mcve_deleteresponse -- Delete specified transaction from MCVE_CONN structure

Description

bool **mcve_deleteresponse** (resource conn, int identifier)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_deletetrans

(PHP 4 >= 4.3.0, PHP 5)

mcve_deletetrans -- Delete specified transaction from MCVE_CONN structure

Description

bool **mcve_deletetrans** (resource conn, int identifier)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_deleteusersetup

(PHP 4 >= 4.2.0, PHP 5)

mcve_deleteusersetup -- Deallocate data associated with usersetup structure

Description

void **mcve_deleteusersetup** (resource usersetup)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_deluser

(PHP 4 >= 4.2.0, PHP 5)

mcve_deluser -- Delete an MCVE user account

Description

int **mcve_deluser** (resource conn, string admin_password, string username)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_destroyconn

(PHP 4 >= 4.2.0, PHP 5)

mcve_destroyconn -- Destroy the connection and MCVE_CONN structure

Description

void **mcve_destroyconn** (resource conn)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_destroyengine

(PHP 4 >= 4.2.0, PHP 5)

mcve_destroyengine -- Free memory associated with IP/SSL connectivity

Description

void **mcve_destroyengine** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_disableuser

(PHP 4 >= 4.2.0, PHP 5)

mcve_disableuser -- Disable an active MCVE user account

Description

int **mcve_disableuser** (resource conn, string admin_password, string username)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_edituser

(PHP 4 >= 4.2.0, PHP 5)

mcve_edituser -- Edit MCVE user using usersetup structure

Description

int **mcve_edituser** (resource conn, string admin_password, int usersetup)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_enableuser

(PHP 4 >= 4.2.0, PHP 5)

mcve_enableuser -- Enable an inactive MCVE user account

Description

int **mcve_enableuser** (resource conn, string admin_password, string username)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_force

(PHP 4 >= 4.2.0, PHP 5)

mcve_force -- Send a FORCE to MCVE (typically, a phone-authorization)

Description

int **mcve_force** (resource conn, string username, string password, string trackdata, string account, string expdate, float amount, string authcode, string comments, string clerkid, string stationid, int ptrannum)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_getcell

(PHP 4 >= 4.2.0, PHP 5)

mcve_getcell -- Get a specific cell from a comma delimited response by column name

Description

string **mcve_getcell** (resource conn, int identifier, string column, int row)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_getcellbynum

(PHP 4 >= 4.2.0, PHP 5)

mcve_getcellbynum -- Get a specific cell from a comma delimited response by column number

Description

string **mcve_getcellbynum** (resource conn, int identifier, int column, int row)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_getcommadelimited

(PHP 4 >= 4.2.0, PHP 5)

mcve_getcommadelimited -- Get the RAW comma delimited data returned from MCVE

Description

string **mcve_getcommadelimited** (resource conn, int identifier)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_getheader

(PHP 4 >= 4.2.0, PHP 5)

mcve_getheader -- Get the name of the column in a comma-delimited response

Description

string **mcve_getheader** (resource conn, int identifier, int column_num)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_getuserarg

(PHP 4 >= 4.2.0, PHP 5)

mcve_getuserarg -- Grab a value from usersetup structure

Description

string **mcve_getuserarg** (resource usersetup, int argtype)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_getuserparam

(PHP 4 >= 4.3.0, PHP 5)

mcve_getuserparam -- Get a user response parameter

Description

string **mcve_getuserparam** (resource conn, int identifier, int key)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_gft

(PHP 4 >= 4.2.0, PHP 5)

mcve_gft -- Audit MCVE for Failed transactions

Description

int **mcve_gft** (resource conn, string username, string password, int type, string account, string clerkid, string stationid, string comments, int ptrannum, string startdate, string enddate)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_gl

(PHP 4 >= 4.2.0, PHP 5)

mcve_gl -- Audit MCVE for settled transactions

Description

int **mcve_gl** (int conn, string username, string password, int type, string account, string batch, string clerkid, string stationid, string comments, int ptrannum, string startdate, string enddate)

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_gut

(PHP 4 >= 4.2.0, PHP 5)

mcve_gut -- Audit MCVE for Unsettled Transactions

Description

int **mcve_gut** (resource conn, string username, string password, int type, string account, string clerkid, string stationid, string comments, int ptrannum, string startdate, string enddate)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_initconn

(PHP 4 >= 4.2.0, PHP 5)

mcve_initconn -- Create and initialize an MCVE_CONN structure

Description

resource **mcve_initconn** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_initengine

(PHP 4 >= 4.2.0, PHP 5)

mcve_initengine -- Ready the client for IP/SSL Communication

Description

int **mcve_initengine** (string location)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_initusersetup

(PHP 4 >= 4.2.0, PHP 5)

mcve_initusersetup -- Initialize structure to store user data

Description

resource **mcve_initusersetup** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_iscommadelimited

(PHP 4 >= 4.2.0, PHP 5)

mcve_iscommadelimited -- Checks to see if response is comma delimited

Description

int **mcve_iscommadelimited** (resource conn, int identifier)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_liststats

(PHP 4 >= 4.2.0, PHP 5)

mcve_liststats -- List statistics for all users on MCVE system

Description

int **mcve_liststats** (resource conn, string admin_password)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_listusers

(PHP 4 >= 4.2.0, PHP 5)

mcve_listusers -- List all users on MCVE system

Description

int **mcve_listusers** (resource conn, string admin_password)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_maxconntimeout

(PHP 4 >= 4.3.0, PHP 5)

mcve_maxconntimeout -- The maximum amount of time the API will attempt a connection to MCVE

Description

bool **mcve_maxconntimeout** (resource conn, int secs)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_monitor

(PHP 4 >= 4.2.0, PHP 5)

mcve_monitor -- Perform communication with MCVE (send/receive data) Non-blocking

Description

int **mcve_monitor** (resource conn)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_numcolumns

(PHP 4 >= 4.2.0, PHP 5)

mcve_numcolumns -- Number of columns returned in a comma delimited response

Description

int **mcve_numcolumns** (resource conn, int identifier)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_numrows

(PHP 4 >= 4.2.0, PHP 5)

mcve_numrows -- Number of rows returned in a comma delimited response

Description

int **mcve_numrows** (resource conn, int identifier)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_override

(PHP 4 >= 4.2.0, PHP 5)

mcve_override -- Send an OVERRIDE to MCVE

Description

int **mcve_override** (resource conn, string username, string password, string trackdata, string account, string expdate, float amount, string street, string zip, string cv, string comments, string clerkid, string stationid, int ptrannum)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_parsecommadelimited

(PHP 4 >= 4.2.0, PHP 5)

mcve_parsecommadelimited -- Parse the comma delimited response so mcve_getcell, etc will work

Description

int **mcve_parsecommadelimited** (resource conn, int identifier)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_ping

(PHP 4 >= 4.3.0, PHP 5)

mcve_ping -- Send a ping request to MCVE

Description

int **mcve_ping** (resource conn)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_preauth

(PHP 4 >= 4.2.0, PHP 5)

mcve_preauth -- Send a PREAUTHORIZATION to MCVE

Description

int **mcve_preauth** (resource conn, string username, string password, string trackdata, string account, string expdate, float amount, string street, string zip, string cv, string comments, string clerkid, string stationid, int ptrannum)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_preauthcompletion

(PHP 4 >= 4.2.0, PHP 5)

mcve_preauthcompletion -- Complete a PREAUTHORIZATION, ready it for settlement

Description

int **mcve_preauthcompletion** (resource conn, string username, string password, float finalamount, int sid, int ptrannum)

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_qc

(PHP 4 >= 4.2.0, PHP 5)

mcve_qc -- Audit MCVE for a list of transactions in the outgoing queue

Description

int **mcve_qc** (resource conn, string username, string password, string clerkid, string stationid, string comments, int ptrannum)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_responseparam

(PHP 4 >= 4.3.0, PHP 5)

mcve_responseparam -- Get a custom response parameter

Description

string **mcve_responseparam** (resource conn, int identifier, string key)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_return

(PHP 4 >= 4.2.0, PHP 5)

mcve_return -- Issue a RETURN or CREDIT to MCVE

Description

int **mcve_return** (int conn, string username, string password, string trackdata, string account, string expdate, float amount, string comments, string clerkid, string stationid, int ptrannum)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_returncode

(PHP 4 >= 4.2.0, PHP 5)

mcve_returncode -- Grab the exact return code from the transaction

Description

int **mcve_returncode** (resource conn, int identifier)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_returnstatus

(PHP 4 >= 4.2.0, PHP 5)

mcve_returnstatus -- Check to see if the transaction was successful

Description

int **mcve_returnstatus** (resource conn, int identifier)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_sale

(PHP 4 >= 4.2.0, PHP 5)

mcve_sale -- Send a SALE to MCVE

Description

int **mcve_sale** (resource conn, string username, string password, string trackdata, string account, string expdate, float amount, string street, string zip, string cv, string comments, string clerkid, string stationid, int ptrannum)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_setblocking

(PHP 4 >= 4.3.0, PHP 5)

mcve_setblocking -- Set blocking/non-blocking mode for connection

Description

int **mcve_setblocking** (resource conn, int tf)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_setdropfile

(PHP 4 >= 4.2.0, PHP 5)

mcve_setdropfile -- Set the connection method to Drop-File

Description

int **mcve_setdropfile** (resource conn, string directory)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_setip

(PHP 4 >= 4.2.0, PHP 5)

mcve_setip -- Set the connection method to IP

Description

int **mcve_setip** (resource conn, string host, int port)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_setssl_files

(PHP 4 >= 4.3.3, PHP 5)

mcve_setssl_files -- Set certificate key files and certificates if server requires client certificate verification

Description

int **mcve_setssl_files** (string sslkeyfile, string sslcertfile)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_setssl

(PHP 4 >= 4.2.0, PHP 5)

mcve_setssl -- Set the connection method to SSL

Description

int **mcve_setssl** (resource conn, string host, int port)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_settimeout

(PHP 4 >= 4.2.0, PHP 5)

mcve_settimeout -- Set maximum transaction time (per trans)

Description

int **mcve_settimeout** (resource conn, int seconds)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_settle

(PHP 4 >= 4.2.0, PHP 5)

mcve_settle -- Issue a settlement command to do a batch deposit

Description

int **mcve_settle** (resource conn, string username, string password, string batch)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_text_avs

(PHP 4 >= 4.3.0, PHP 5)

mcve_text_avs -- Get a textual representation of the return_avs

Description

string **mcve_text_avs** (string code)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_text_code

(PHP 4 >= 4.3.0, PHP 5)

mcve_text_code -- Get a textual representation of the return_code

Description

string **mcve_text_code** (string code)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_text_cv

(PHP 4 >= 4.3.0, PHP 5)

mcve_text_cv -- Get a textual representation of the return_cv

Description

string **mcve_text_cv** (int code)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_transactionauth

(PHP 4 >= 4.2.0, PHP 5)

mcve_transactionauth -- Get the authorization number returned for the transaction (alpha-numeric)

Description

string **mcve_transactionauth** (resource conn, int identifier)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_transactionavs

(PHP 4 >= 4.2.0, PHP 5)

mcve_transactionavs -- Get the Address Verification return status

Description

int **mcve_transactionavs** (resource conn, int identifier)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_transactionbatch

(PHP 4 >= 4.2.0, PHP 5)

mcve_transactionbatch -- Get the batch number associated with the transaction

Description

int **mcve_transactionbatch** (resource conn, int identifier)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_transactioncv

(PHP 4 >= 4.2.0, PHP 5)

mcve_transactioncv -- Get the CVC2/CVV2/CID return status

Description

int **mcve_transactioncv** (resource conn, int identifier)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_transactionid

(PHP 4 >= 4.2.0, PHP 5)

mcve_transactionid -- Get the unique system id for the transaction

Description

int **mcve_transactionid** (resource conn, int identifier)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_transactionitem

(PHP 4 >= 4.2.0, PHP 5)

mcve_transactionitem -- Get the ITEM number in the associated batch for this transaction

Description

int **mcve_transactionitem** (resource conn, int identifier)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_transactionssent

(PHP 4 >= 4.2.0, PHP 5)

mcve_transactionssent -- Check to see if outgoing buffer is clear

Description

int **mcve_transactionssent** (resource conn)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_transactiontext

(PHP 4 >= 4.2.0, PHP 5)

mcve_transactiontext -- Get verbiage (text) return from MCVE or processing institution

Description

string **mcve_transactiontext** (resource conn, int identifier)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_transinqueue

(PHP 4 >= 4.2.0, PHP 5)

mcve_transinqueue -- Number of transactions in client-queue

Description

int **mcve_transinqueue** (resource conn)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_transnew

(PHP 4 >= 4.3.0, PHP 5)

mcve_transnew -- Start a new transaction

Description

int **mcve_transnew** (resource conn)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_transparam

(PHP 4 >= 4.3.0, PHP 5)

mcve_transparam -- Add a parameter to a transaction

Description

int **mcve_transparam** (resource conn, int identifier, int key)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_transsend

(PHP 4 >= 4.3.0, PHP 5)

mcve_transsend -- Finalize and send the transaction

Description

int **mcve_transsend** (resource conn, int identifier)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_ub

(PHP 4 >= 4.2.0, PHP 5)

mcve_ub -- Get a list of all Unsettled batches

Description

int **mcve_ub** (resource conn, string username, string password)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_uwait

(PHP 4 >= 4.3.0, PHP 5)

mcve_uwait -- Wait x microsecs

Description

int **mcve_uwait** (int microsecs)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_verifyconnection

(PHP 4 >= 4.3.0, PHP 5)

mcve_verifyconnection -- Set whether or not to PING upon connect to verify connection

Description

bool **mcve_verifyconnection** (resource conn, int tf)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_verifysslcert

(PHP 4 >= 4.3.0, PHP 5)

mcve_verifysslcert -- Set whether or not to verify the server ssl certificate

Description

bool **mcve_verifysslcert** (resource conn, int tf)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mcve_void

(PHP 4 >= 4.2.0, PHP 5)

mcve_void -- VOID a transaction in the settlement queue

Description

int **mcve_void** (resource conn, string username, string password, int sid, int ptrannum)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

LXIX. Memcache Functions

Introducción

Memcache module provides handy procedural and object oriented interface to memcached, highly effective caching daemon, which was especially designed to decrease database load in dynamic web applications.

More information about memcached can be found at http://www.danga.com/memcached/.

Requirimientos

This module uses functions of zlib to support on-the-fly data compression. Zlib is required to install this module.

PHP 4.3.3 or newer is required to use the memcache extension.

Instalación

Esta extension PECL no esta ligada a PHP. Mas informacion sobre nuevos lanzamientos, descargas ficheros de fuentes, informacion sobre los responsables asi como un 'CHANGELOG', se puede encontrar aqui: http://pecl.php.net/package/memcache.

In order to use these functions you must compile PHP with MemCache support by using the *--with-memcache[=DIR]* option.

Windows users will enable `php_memcache.dll` inside of `php.ini` in order to use these functions. Podeis descargar esta DLL de las extensiones PECL desde la pagina PHP Downloads o desde http://snaps.php.net/.

Constantes predefinidas

Tabla 1. MemCache Constants

Name	Description
MEMCACHE_COMPRESSED (integer)	Used to turn on-the-fly data compression on with **Memcache::set()**, **Memcache::add()**, y **Memcache::replace()**.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

There is only one resource type used in memcache module - it's the link identifier for a cache server connection.

Ejemplos

Ejemplo 1. memcache extension overview example

```
<?php

$memcache = new Memcache;
$memcache->connect('localhost', 11211) or die ("Could not connect");

$version = $memcache->getVersion();
echo "Server's version: ".$version."<br/>\n";

$tmp_object = new stdClass;
$tmp_object->str_attr = 'test';
$tmp_object->int_attr = 123;

$memcache->set('key', $tmp_object, false, 10) or die ("Failed to save data at the serve
echo "Store data in the cache (data will expire in 10 seconds)<br/>\n";

$get_result = $memcache->get('key');
echo "Data from the cache:<br/>\n";

var_dump($get_result);

?>
```

In the above example, an object is being saved in the cache and then retrieved back. Object and other non-scalar types are serialized before saving, so it's impossible to store resources (i.e. connection identifiers and others) in the cache.

Tabla de contenidos

Memcache::increment -- Increment item's value
Memcache::pconnect -- Open memcached server persistent connection
Memcache::replace -- Replace value of the existing item
Memcache::set -- Store data at the server

Memcache::add

(no version information, might be only in CVS)

Memcache::add -- Add an item to the server

Description

bool **Memcache::add** (string key, mixed var [, int flag [, int expire]])

Memcache::add() stores variable *var* with *key* only if such key doesn't exist at the server yet. **Memcache::add()** returns **FALSE** if such key already exist. For the rest **Memcache::add()** behaves similarly to **Memcache::set()**.

Also you can use **memcache_add()** function. See example below.

Ejemplo 1. Memcache::add() example

```
<?php

$memcache_obj = memcache_connect("localhost", 11211);

/* procedural API */
memcache_add($memcache_obj, 'var_key', 'test variable', false, 30);

/* OO API */
$memcache_obj->add('var_key', 'test variable', false, 30);

?>
```

Memcache::add() returns **TRUE** on success or **FALSE** on failure.

See also **Memcache::set()**, **Memcache::replace()**.

Memcache::close

(no version information, might be only in CVS)

Memcache::close -- Close memcached server connection

Description

bool **Memcache::close** (void)

Memcache::close() closes connection to memcached server. This function doesn't close persistent connections, which are closed only during web-server shutdown/restart.

Also you can use **memcache_close()** function. See example below.

Ejemplo 1. Memcache::close() example

```php
<?php

/* procedural API */
$memcache_obj = memcache_connect('memcache_host', 11211);
/*
do something here ..
*/
memcache_close($memcache_obj);

/* OO API */
$memcache_obj = new Memcache;
$memcache_obj->connect('memcache_host', 11211);
/*
do something here ..
*/
$memcache_obj->close();

?>
```

Memcache::close() returns **FALSE** if an error occured.

See also **Memcache::connect()**, **Memcache::pconnect()**.

Memcache::connect

(no version information, might be only in CVS)

Memcache::connect -- Open memcached server connection

Description

bool **Memcache::connect** (string host [, int port [, int timeout]])

Memcache::connect() establishes a connection to the memcached server. Parameters *host* and *port* point to the host and port, where memcached is listening for connections. Parameter *port* is optional, it's default value is 11211. Also you can define a *timeout*, which will be used when connecting to the daemon. Think twice before changing the default value - you can loose all the advantages of caching if your connection is too slow.

The connection, which was opened using **Memcache::connect()** will be automatically closed at the end of script execution. Also you can close it with **Memcache::close()**.

Also you can use **memcache_connect()** function. See example below.

Ejemplo 1. Memcache::connect() example

```php
<?php

/* procedural API */

$memcache_obj = memcache_connect('memcache_host', 11211);

/* OO API */

$memcache = new Memcache;
$memcache->connect('memcache_host', 11211);

?>
```

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

See also **Memcache::pconnect()** and **Memcache::close()**.

memcache_debug

(no version information, might be only in CVS)

memcache_debug -- Turn debug output on/off

Description

bool **memcache_debug** (int on_off)

memcache_debug() turns on debug output if parameter *on_off* is equal to 1 and turns off if it's 0.

> **Nota: memcache_debug()** is accessible only if PHP was built with --enable-debug option and always returns **`TRUE`** in this case. Otherwise, this function has no effect and always returns **`FALSE`**.

Memcache::decrement

(no version information, might be only in CVS)

Memcache::decrement -- Decrement item's value

Description

int **Memcache::decrement** (string key [, int value])

Memcache::decrement() decrements value of the item by *value*. Similarly to **Memcache::increment()**, current value of the item is being converted to numerical and after that *value* is substracted.

Parameter *value* is optional. It's default is 1.

> **Nota:** New item's value will not be less than zero.

> **Nota:** Do not use **Memcache::decrement()** with item, which was stored compressed, because consequent call to **Memcache::get()** will fail.

Also you can use **memcache_decrement()** function. See example below.

Ejemplo 1. Memcache::decrement() example

```
<?php

/* procedural API */
$memcache_obj = memcache_connect('memcache_host', 11211);
/* decrement item by 2 */
$new_value = memcache_decrement($memcache_obj, 'test_item', 2);

/* OO API */
$memcache_obj = new Memcache;
$memcache_obj->connect('memcache_host', 11211);
/* decrement item by 3 */
$new_value = $memcache_obj->decrement('test_item', 3);
?>
```

Memcache::decrement() *does not* create an item if it didn't exist.

Memcache::decrement() returns item's new value on success or **FALSE** on failure.

See also **Memcache::increment()**, **Memcache::replace()**.

Memcache::delete

(no version information, might be only in CVS)

Memcache::delete -- Delete item from the server

Description

bool **Memcache::delete** (string key [, int timeout])

Memcache::delete() deletes item with the *key*. If parameter *timeout* is specified, the item will expire after *timeout* seconds.

Also you can use **memcache_delete()** function. See example below.

Ejemplo 1. Memcache::delete() example

```
<?php

/* procedural API */
$memcache_obj = memcache_connect('memcache_host', 11211);

/* after 10 seconds item will be deleted by the server */
memcache_delete('key_to_delete', 10);

/* OO API */
$memcache_obj = new Memcache;
$memcache_obj->connect('memcache_host', 11211);

$memcache_obj->delete('key_to_delete', 10);

?>
```

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Memcache::flush

(no version information, might be only in CVS)

Memcache::flush -- Flush all existing items at the server

Description

bool **Memcache::flush** (void)

Memcache::flush() immediately invalidates all existing items. **Memcache::flush()** doesn't actually free any resources, it only marks all the items as expired, so occupied memory will be overwritten by new items.

Also you can use **memcache_flush()** function. See example below.

Ejemplo 1. Memcache::flush() example

```
<?php

/* procedural API */
$memcache_obj = memcache_connect('memcache_host', 11211);

memcache_flush($memcache_obj);

/* OO API */

$memcache_obj = new Memcache;
$memcache_obj->connect('memcache_host', 11211);

$memcache_obj->flush();

?>
```

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Memcache::get

(no version information, might be only in CVS)

Memcache::get -- Retrieve item from the server

Description

mixed **Memcache::get** (string key)

mixed **Memcache::get** (array keys)

Memcache::get() returns previously stored data if an item with such *key* exists on the server at this moment.

Ejemplo 1. Memcache::get() example

```
<?php

/* procedural API */
$memcache_obj = memcache_connect('memcache_host', 11211);
$var = memcache_get($memcache_obj, 'some_key');

/* OO API */
$memcache_obj = new Memcache;
$memcache_obj->connect('memcache_host', 11211);
$var = $memcache_obj->get('some_key');

/*
You also can use array of keys as a parameter.
If such item wasn't found at the server, the result
array simply will not include such key.
*/

/* procedural API */
$memcache_obj = memcache_connect('memcache_host', 11211);
$var = memcache_get($memcache_obj, Array('some_key', 'another_key'));

/* OO API */
$memcache_obj = new Memcache;
$memcache_obj->connect('memcache_host', 11211);
$var = $memcache_obj->get(Array('some_key', 'second_key'));

?>
```

Memcache::get() returns **FALSE** on failure or if such *key* was not found.

Memcache::getStats

(no version information, might be only in CVS)

Memcache::getStats -- Get statistics of the server

Description

array **Memcache::getStats** (void)

Memcache::getStats() returns an associative array with server's statistics. Array keys correspond to stats parameters and values to parameter's values.

Also you can use **memcache_get_stats()** function.

Memcache::getStats() returns **FALSE** in case of an error.

See also **Memcache::getVersion()**.

Memcache::getVersion

(no version information, might be only in CVS)

Memcache::getVersion -- Return version of the server

Description

string **Memcache::getVersion** (void)

Memcache::getVersion() returns a string with server's version number.

Also you can use **memcache_get_version()** function. See example below.

Ejemplo 1. Memcache::getVersion() example

```
<?php

/* procedural API */
$memcache_obj = memcache_connect('memcache_host', 11211);

echo memcache_get_version($memcache_obj);

/* OO API */
$memcache_obj = new Memcache;
echo $memcache_obj->getVersion();

?>
```

Memcache::getVersion() returns **FALSE** if an error occured.

See also **Memcache::getStats()**.

Memcache::increment

(no version information, might be only in CVS)

Memcache::increment -- Increment item's value

Description

int **Memcache::increment** (string key [, int value])

Memcache::increment() increments value of the item on the specified *value*. If item with key *key* was not numeric and cannot be converted to number, it will change it's value to *value*.

Parameter *value* is optional. It's default value is 1.

> **Nota:** Do not use **Memcache::increment()** with item, which was stored compressed, because consequent call to **Memcache::get()** will fail.

Also you can use **memcache_increment()** function. See example below.

Ejemplo 1. Memcache::increment() example

```
<?php

/* procedural API */
$memcache_obj = memcache_connect('memcache_host', 11211);
/* increment counter by 2 */
$current_value = memcache_increment($memcache_obj, 'counter', 2);

/* OO API */
$memcache_obj = new Memcache;
$memcache_obj->connect('memcache_host', 11211);
/* increment counter by 3 */
$current_value = $memcache_obj->increment('counter', 3);

?>
```

Memcache::increment() returns new item's value on success or **FALSE** on failure.

Memcache::increment() *does not* create an item if it didn't exist.

See also **Memcache::decrement()**, **Memcache::replace()**.

Memcache::pconnect

(no version information, might be only in CVS)

Memcache::pconnect -- Open memcached server persistent connection

Description

bool **Memcache::pconnect** (string host [, int port [, int timeout]])

Memcache::pconnect() is similar to **Memcache::connect()** with the difference, that the connection it establishes is persistent. This connection is not closed after the end of script execution and by **Memcache::close()** function.

Also you can use **memcache_pconnect()** function. See example below.

Ejemplo 1. Memcache::pconnect() example

```
<?php

/* procedural API */
$memcache_obj = memcache_pconnect('memcache_host', 11211);

/* OO API */

$memcache_obj = new Memcache;
$memcache_obj->pconnect('memcache_host', 11211);

?>
```

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

See also **Memcache::connect()**.

Memcache::replace

(no version information, might be only in CVS)

Memcache::replace -- Replace value of the existing item

Description

bool **Memcache::replace** (string key, mixed var [, int flag [, int expire]])

Memcache::replace() should be used to replace value of existing item with *key*. In case if item with such key doesn't exists, **Memcache::replace()** returns **FALSE**. For the rest **Memcache::replace()** behaves similarly to **Memcache::set()**.

Also you can use **memcache_replace()** function. See example below.

Ejemplo 1. Memcache::replace() example

```php
<?php

$memcache_obj = memcache_connect('memcache_host', 11211);

/* procedural API */
memcache_replace($memcache_obj, "test_key", "some variable", false, 30);

/* OO API */
$memcache_obj->replace("test_key", "some variable", false, 30);

?>
```

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

See also **Memcache::set()**, **Memcache::add()**.

Memcache::set

(no version information, might be only in CVS)

Memcache::set -- Store data at the server

Description

bool **Memcache::set** (string key, mixed var [, int flag [, int expire]])

Memcache::set() stores an item *var* with *key* on the memcached server. Parameter *expire* is expiration time in seconds. If it's 0, the item never expires (but memcached server doesn't guarantee this item to be stored all the time, it could be deleted from the cache to make place for other items).

You can use **MEMCACHE_COMPRESSED** constant as *flag* value if you want to use on-the-fly compression (uses zlib).

Also you can use **memcache_set()** function. See example below.

> **Nota:** Remember that resource variables (i.e. file and connection descriptors) cannot be stored in the cache, because they cannot be adequately represented in serialized state.

Ejemplo 1. Memcache::set() example

```php
<?php
/* procedural API */

/* connect to memcached server */
$memcache_obj = memcache_connect('memcache_host', 11211);

/*
set value of item with key 'var_key'
using 0 as flag value, compression is not used
expire time is 30 seconds
*/
memcache_set($memcache_obj, 'var_key', 'some variable', 0, 30);

echo memcache_get($memcache_obj, 'var_key');

?>
```

Ejemplo 2. Memcache::set() example

```
<?php
/* OO API */

$memcache_obj = new Memcache;

/* connect to memcached server */
$memcache->connect('memcache_host', 11211);

/*
set value of item with key 'var_key', using on-the-fly compression
expire time is 50 seconds
*/
$memcache_obj->set('var_key', 'some really big variable', MEMCACHE_COMPRESSED, 50);

echo $memcache_obj->get('var_key');

?>
```

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

See also **Memcache::add()**, **Memcache::replace()**.

LXX. Funciones Mhash

Introducción

Estas funciones tienen el propósito de trabajar con mhash. Mhash puede ser usado para crear sumas de verificación, resúmenes de mensajes, códigos de autenticación de mensajes, y más.

Esta es una interfaz con la biblioteca mhash. mhash soporta una amplia variaded de algoritmos hash como MD5, SHA1, GOST, y muchos otros. Para una lista completa de resúmenes criptográficos soportados, refiérase a la documentación de mhash. La regla general es que puede acceder al algoritmo hash desde PHP con MHASH_NOMBRE_DEL_HASH. Por ejemplo, para acceder a TIGER, use la constante PHP MHASH_TIGER.

Requirimientos

Para usar la extensión, descargue la distribución de mhash desde su sitio web y siga las instrucciones de instalación incluidas.

Instalación

Necesita compilar PHP con el parámetro *--with-mhash[=DIR]* para habilitar esta extensión. DIR es el directorio de instalación de mhash.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

A continuación se encuentra una lista de resúmenes criptográficos soportados en la actualidad por mhash. Si un mecanismo de resumen no se encuentra listado aquí, pero es listado como soportado por mhash, puede asumir con seguridad que esta documentación se encuentra desactualizada.

- **MHASH_MD5**
- **MHASH_SHA1**
- **MHASH_HAVAL256**
- **MHASH_HAVAL192**
- **MHASH_HAVAL160**
- **MHASH_HAVAL128**
- **MHASH_RIPEMD160**
- **MHASH_GOST**
- **MHASH_TIGER**
- **MHASH_CRC32**
- **MHASH_CRC32B**

Ejemplos

Ejemplo 1. Calcular el resumen MD5 y hmac e imprimirlo como valor hexadecimal

```
<?php
$entrada = "que quisiera a cambio de nada?";
$hash = mhash(MHASH_MD5, $entrada);
echo "El valor hash es ".bin2hex ($hash)."<br />\n";
$hash = mhash(MHASH_MD5, $entrada, "Jefe");
echo "El valor hmac es ".bin2hex ($hash)."<br />\n";
?>
```

Esto producirá:

```
El valor hash es 2386e00b2d014a4b89efb10b0250ac35
El valor hmac es 87367c5cbb5099cf95fcee560d402da4
```

Tabla de contenidos

mhash_count

(PHP 3>= 3.0.9, PHP 4 , PHP 5)

mhash_count -- Obtener el valor mayor del id hash disponible

Descripcion

int **mhash_count** (void)

mhash_count() devuelve el valor mas alto id hash disponible. Los hash estan numerados desde 0 hasta este valor.

Ejemplo 1. Recorriendo todos los hash

```
<?php

$nr = mhash_count();

for($i = 0; $i <= $nr; $i++) {
    echo sprintf("The blocksize of %s is %d\n",
            mhash_get_hash_name($i),
            mhash_get_block_size($i));
}
?>
```

mhash_get_block_size

(PHP 3>= 3.0.9, PHP 4 , PHP 5)

mhash_get_block_size -- Conseguir el tamaño de bloque de el hash especificado

Descripcion

int **mhash_get_block_size** (int hash)

mhash_get_block_size() es usado para obtener el tamaño de un bloque de el *hash* determinado.

mhash_get_block_size() toma un argumento, el *hash* y devuelve el tamaño en bytes o **FALSE**, si el *hash* no existe.

mhash_get_hash_name

(PHP 3>= 3.0.9, PHP 4 , PHP 5)

mhash_get_hash_name -- Conseguir el nombre de un hash especifico

Descripcion

string **mhash_get_hash_name** (int hash)

mhash_get_hash_name() es usado para conseguir el nombre de el hash determinado.

mhash_get_hash_name() toma el id del hash como un argumento y devuelve el nombre de el hash o **FALSE**, si el hash no existe.

Ejemplo 1. mhash_get_hash_name example

```
<?php
$hash = MHASH_MD5;

print mhash_get_hash_name($hash);
?>
```

El ejemplo anterior mostrara:

```
MD5
```

mhash_keygen_s2k

(PHP 4 >= 4.0.4, PHP 5)

mhash_keygen_s2k -- Genera una llave

Descripción

string **mhash_keygen_s2k** (int hash, string contrasenya, string sal, int bytes)

mhash_keygen_s2k() genera una llave con una longitud indicada por *bytes*, a partir de una contraseña entregada por el usuario. Este es el algoritmo con sal S2K, tal y como se encuentra especificado en el documento OpenPGP (RFC 2440). Ese algoritmo usará el algoritmo *hash* especificado para crear la llave. La *sal* debe ser diferente y suficientemente aleatoria para cada llave que genere para crear diferentes llaves. La sal debe conocerse cuando chequee las llaves, de modo que es una buena idea adicionarle la llave al final. La sal tiene un tamaño fijo de 8 bytes y será rellenada con ceros si usted suple menos bytes.

Tenga en mente que las contraseñas entregadas por el usuario no son realmente apropiadas para ser usadas como llaves en algoritmos criptográficos, ya que los usuarios normalmente eligen llaves que pueden escribir con el teclado. Estas contreseñas usan solo 6 o 7 bits por caracter (o menos). Es bastante recomendable usar algún tipo de transformación (como esta función) sobre la llave entregada por el usuario.

mhash

(PHP 3>= 3.0.9, PHP 4 , PHP 5)

mhash -- Calcular el hash

Descripcion

string **mhash** (int hash, string data)

mhash() aplica una funcion hash especificada por *hash* a *data* y devuelve el valor hash resultante (tambien llamdo digest).

LXXI. Funciones Mimetype

Introducción

Aviso
Esta extensión ha sido marcada como obsoleta, ya que la extensión PECL fileinfo ofrece la misma funcionalidad (y más) de una forma mucho más clara.

Las funciones de este módulo intentan adivinar el tipo de contenido y la codificación de un archivo, mirando ciertas secuencias de bytes *mágicas* en posiciones específicas dentro del archivo. Aunque éste no es un enfoque completamente seguro, la heurística usada cumple un muy buen trabajo.

Esta extensión es derivada del módulo de Apache mod_mime_magic, el cual a su vez está basado en el comando `file`, administrado por Ian F. Darwin. Vea el código fuente para consultar más datos históricos y la información de copyright.

Requirimientos

No se necesitan bibliotecas externas para construir esta extensión

---

Instalación

Debe compilar PHP con la opción de configuración *--with-mime-magic* para contar con soporte para funciones mime-type. La extensión necesita una copia del archivo `magic` simplificado, que es distribuido con el servidor web Apache.

> **Nota:** La opción de configuración ha cambiado de *--enable-mime-magic* a *--with-mime-magic* a partir de PHP 4.3.2.
>
> **Nota:** Esta extensión no es capaz de gestionar el archivo `magic` completamente decorado que generalmente viene con las distribuciones normales de Linux, y es utilizado por lo general con versiones recientes del comando `file`.
>
> **Nota para Usuarios de Win32:** Para usar este módulo en un entorno Windows, debe definir la ruta hacia el archivo `magic.mime` distribuido en su `php.ini`.
>
> **Ejemplo 1. Definición de la ruta hacia `magic.mime`**
>
> ```
> mime_magic.magicfile ="$PHP_INSTALL_DIR\magic.mime"
> ```
>
> Recuerde sustituir el valor `$PHP_INSTALL_DIR` con su ruta real hacia PHP en el anterior ejemplo. Por ejemplo, `c:\php`

Configuración en tiempo de ejecución

El comportamiento de estas funciones está afectado por los valores definidos en `php.ini`.

Tabla 1. Opciones de configuración de Mimetype

Nombre	Predeterminado	Modificable
mime_magic.magicfile	"/usr/share/misc/magic.mime"	PHP_INI_SYSTEM

Para más detalles sobre las constantes PHP_INI_* y su definición, vea **ini_set()**.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Esta extensión no tiene ninguna constante definida.

Tabla de contenidos

mime_content_type

(PHP 4 >= 4.3.0, PHP 5)

mime_content_type -- Detectar el tipo de contenido MIME de un archivo

Descripción

string **mime_content_type** (string nombre_archivo)

Devuelve el tipo de contenido MIME de un archivo, tal y como sea determinado por medio del uso de información tomada del archivo `magic.mime`. Los tipos de contenido son devueltos en formato MIME, como *text/plain* o *application/octet-stream*.

Ejemplo 1. Ejemplo de mime_content_type()

```
<?php
echo mime_content_type('php.gif') . "\n";
echo mime_content_type('test.php');
?>
```

El anterior ejemplo producirá la salida:

```
image/gif
text/plain
```

LXXII. Ming functions for Flash

Aviso
Esta extensión es *EXPERIMENTAL*. Esto significa que el comportamiento de esta extensión, los nombre de sus funciones y en definitiva TODO lo documentado sobre esta extensión, puede cambiar en una futura versión de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Introducción

First of all: Ming is not an acronym. Ming is an open-source (LGPL) library which allows you to create SWF ("Flash") format movies. Ming supports almost all of Flash 4's features, including: shapes, gradients, bitmaps (pngs and jpegs), morphs ("shape tweens"), text, buttons, actions, sprites ("movie clips"), streaming mp3, and color transforms --the only thing that's missing is sound events.

Note that all values specifying length, distance, size, etc. are in "twips", twenty units per pixel. That's pretty much arbitrary, though, since the player scales the movie to whatever pixel size is specified in the embed/object tag, or the entire frame if not embedded.

Ming offers a number of advantages over the existing PHP/libswf module. You can use Ming anywhere you can compile the code, whereas libswf is closed-source and only available for a few platforms, Windows not one of them. Ming provides some insulation from the mundane details of the SWF file format, wrapping the movie elements in PHP objects. Also, Ming is still being maintained; if there's a feature that you want to see, just let us know ming@opaque.net.

Ming was added in PHP 4.0.5.

Requirimientos

To use Ming with PHP, you first need to build and install the Ming library. Source code and installation instructions are available at the Ming home page: http://ming.sourceforge.net/ along with examples, a small tutorial, and the latest news.

Download the ming archive. Unpack the archive. Go in the Ming directory. make. make install.

This will build `libming.so` and install it into `/usr/lib/`, and copy `ming.h` into `/usr/include/`. Edit the *PREFIX*= line in the `Makefile` to change the installation directory.

Instalación

Ejemplo 1. built into PHP (Unix)

```
mkdir <phpdir>/ext/ming
cp php_ext/* <phpdir>/ext/ming
cd <phpdir>
./buildconf
./configure --with-ming <other config options>
```

Build and install PHP as usual, restart web server if necessary.

Now either just add *extension=php_ming.so* to your `php.ini` file, or put *dl('php_ming.so');* at the head of all of your Ming scripts.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

`SWFBUTTON_HIT` (integer)

`SWFBUTTON_DOWN` (integer)

`SWFBUTTON_OVER` (integer)

`SWFBUTTON_UP` (integer)

`SWFBUTTON_MOUSEUPOUTSIDE` (integer)

`SWFBUTTON_DRAGOVER` (integer)

`SWFBUTTON_DRAGOUT` (integer)

`SWFBUTTON_MOUSEUP` (integer)

`SWFBUTTON_MOUSEDOWN` (integer)

SWFBUTTON_MOUSEOUT (integer)

SWFBUTTON_MOUSEOVER (integer)

SWFFILL_RADIAL_GRADIENT (integer)

SWFFILL_LINEAR_GRADIENT (integer)

SWFFILL_TILED_BITMAP (integer)

SWFFILL_CLIPPED_BITMAP (integer)

SWFTEXTFIELD_HASLENGTH (integer)

SWFTEXTFIELD_NOEDIT (integer)

SWFTEXTFIELD_PASSWORD (integer)

SWFTEXTFIELD_MULTILINE (integer)

SWFTEXTFIELD_WORDWRAP (integer)

SWFTEXTFIELD_DRAWBOX (integer)

SWFTEXTFIELD_NOSELECT (integer)

SWFTEXTFIELD_HTML (integer)

SWFTEXTFIELD_ALIGN_LEFT (integer)

SWFTEXTFIELD_ALIGN_RIGHT (integer)

SWFTEXTFIELD_ALIGN_CENTER (integer)

SWFTEXTFIELD_ALIGN_JUSTIFY (integer)

SWFACTION_ONLOAD (integer)

SWFACTION_ENTERFRAME (integer)

SWFACTION_UNLOAD (integer)

SWFACTION_MOUSEMOVE (integer)

SWFACTION_MOUSEDOWN (integer)

SWFACTION_MOUSEUP (integer)

SWFACTION_KEYDOWN (integer)

SWFACTION_KEYUP (integer)

SWFACTION_DATA (**integer**)

Clases predefinidas

Estas clases están definidas por esta extensión y estarán disponibles cuando la extensión haya sido compilada dentro de PHP o cargada dinámicamente en tiempo de ejecución

Ming introduces 13 new objects in PHP, all with matching methods and attributes. To use them, you need to know about objects.

swfshape

swffill

swfgradient

swfbitmap

swftext

swftextfield

swffont

swfdisplayitem

swfmovie

swfbutton

swfaction

swfmorph

swfsprite

Tabla de contenidos

SWFbutton -- Creates a new Button
SWFDisplayItem->addColor -- Adds the given color to this item's color transform
SWFDisplayItem->move -- Moves object in relative coordinates
SWFDisplayItem->moveTo -- Moves object in global coordinates
SWFDisplayItem->multColor -- Multiplies the item's color transform
SWFDisplayItem->remove -- Removes the object from the movie
SWFDisplayItem->Rotate -- Rotates in relative coordinates
SWFDisplayItem->rotateTo -- Rotates the object in global coordinates
SWFDisplayItem->scale -- Scales the object in relative coordinates
SWFDisplayItem->scaleTo -- Scales the object in global coordinates
SWFDisplayItem->setDepth -- Sets z-order
SWFDisplayItem->setName -- Sets the object's name
SWFDisplayItem->setRatio -- Sets the object's ratio
SWFDisplayItem->skewX -- Sets the X-skew
SWFDisplayItem->skewXTo -- Sets the X-skew
SWFDisplayItem->skewY -- Sets the Y-skew
SWFDisplayItem->skewYTo -- Sets the Y-skew
SWFDisplayItem -- Creates a new displayitem object
SWFFill->moveTo -- Moves fill origin
SWFFill->rotateTo -- Sets fill's rotation
SWFFill->scaleTo -- Sets fill's scale
SWFFill->skewXTo -- Sets fill x-skew
SWFFill->skewYTo -- Sets fill y-skew
SWFFill -- Loads SWFFill object
swffont->getwidth -- Returns the string's width
SWFFont -- Loads a font definition
SWFGradient->addEntry -- Adds an entry to the gradient list
SWFGradient -- Creates a gradient object
SWFMorph->getshape1 -- Gets a handle to the starting shape
SWFMorph->getshape2 -- Gets a handle to the ending shape
SWFMorph -- Creates a new SWFMorph object
SWFMovie->add -- Adds any type of data to a movie
SWFMovie->nextframe -- Moves to the next frame of the animation
SWFMovie->output -- Dumps your lovingly prepared movie out
swfmovie->remove -- Removes the object instance from the display list
SWFMovie->save -- Saves your movie in a file
SWFMovie->setbackground -- Sets the background color
SWFMovie->setdimension -- Sets the movie's width and height
SWFMovie->setframes -- Sets the total number of frames in the animation
SWFMovie->setrate -- Sets the animation's frame rate
SWFMovie->streammp3 -- Streams a MP3 file
SWFMovie -- Creates a new movie object, representing an SWF version 4 movie
SWFShape->addFill -- Adds a solid fill to the shape
SWFShape->drawCurve -- Draws a curve (relative)
SWFShape->drawCurveTo -- Draws a curve
SWFShape->drawLine -- Draws a line (relative)
SWFShape->drawLineTo -- Draws a line
SWFShape->movePen -- Moves the shape's pen (relative)
SWFShape->movePenTo -- Moves the shape's pen
SWFShape->setLeftFill -- Sets left rasterizing color
SWFShape->setLine -- Sets the shape's line style
SWFShape->setRightFill -- Sets right rasterizing color
SWFShape -- Creates a new shape object
swfsprite->add -- Adds an object to a sprite
SWFSprite->nextframe -- Moves to the next frame of the animation

SWFSprite->remove -- Removes an object to a sprite
SWFSprite->setframes -- Sets the total number of frames in the animation
SWFSprite -- Creates a movie clip (a sprite)
SWFText->addString -- Draws a string
SWFText->getWidth -- Computes string's width
SWFText->moveTo -- Moves the pen
SWFText->setColor -- Sets the current font color
SWFText->setFont -- Sets the current font
SWFText->setHeight -- Sets the current font height
SWFText->setSpacing -- Sets the current font spacing
SWFText -- Creates a new SWFText object
SWFTextField->addstring -- Concatenates the given string to the text field
SWFTextField->align -- Sets the text field alignment
SWFTextField->setbounds -- Sets the text field width and height
SWFTextField->setcolor -- Sets the color of the text field
SWFTextField->setFont -- Sets the text field font
SWFTextField->setHeight -- Sets the font height of this text field font
SWFTextField->setindentation -- Sets the indentation of the first line
SWFTextField->setLeftMargin -- Sets the left margin width of the text field
SWFTextField->setLineSpacing -- Sets the line spacing of the text field
SWFTextField->setMargins -- Sets the margins width of the text field
SWFTextField->setname -- Sets the variable name
SWFTextField->setrightMargin -- Sets the right margin width of the text field
SWFTextField -- Creates a text field object

ming_setcubicthreshold

(PHP 4 >= 4.0.5, PHP 5)

ming_setcubicthreshold -- Set cubic threshold (?)

Description

void **ming_setcubicthreshold** (int threshold)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ming_setscale

(PHP 4 >= 4.0.5, PHP 5)

ming_setscale -- Set scale (?)

Description

void **ming_setscale** (int scale)

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ming_useswfversion

(PHP 4 >= 4.2.0, PHP 5)

ming_useswfversion -- Use SWF version (?)

Description

void **ming_useswfversion** (int version)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

SWFAction

(PHP 4 >= 4.0.5)

SWFAction -- Creates a new Action

Description

SWFAction **swfaction** (string script)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfaction() creates a new Action, and compiles the given script into an SWFAction object.

The script syntax is based on the C language, but with a lot taken out- the SWF bytecode machine is just too simpleminded to do a lot of things we might like. For instance, we can't implement function calls without a tremendous amount of hackery because the jump bytecode has a hardcoded offset value. No pushing your calling address to the stack and returning- every function would have to know exactly where to return to.

So what's left? The compiler recognises the following tokens:

- break
- for
- continue
- if

- else
- do
- while

There is no typed data; all values in the SWF action machine are stored as strings. The following functions can be used in expressions:

time()

Returns the number of milliseconds (?) elapsed since the movie started.

random(seed)

Returns a pseudo-random number in the range 0-seed.

length(expr)

Returns the length of the given expression.

int(number)

Returns the given number rounded down to the nearest integer.

concat(expr, expr)

Returns the concatenation of the given expressions.

ord(expr)

Returns the ASCII code for the given character

chr(num)

Returns the character for the given ASCII code

substr(string, location, length)

Returns the substring of length length at location location of the given string string.

Additionally, the following commands may be used:

duplicateClip(clip, name, depth)

Duplicate the named movie clip (aka sprite). The new movie clip has name name and is at depth depth.

removeClip(expr)

Removes the named movie clip.

trace(expr)

Write the given expression to the trace log. Doubtful that the browser plugin does anything with this.

startDrag(target, lock, [left, top, right, bottom])

Start dragging the movie clip target. The lock argument indicates whether to lock the mouse (?)- use 0 (**FALSE**) or 1 (**TRUE**). Optional parameters define a bounding area for the dragging.

stopDrag()

Stop dragging my heart around. And this movie clip, too.

callFrame(expr)

Call the named frame as a function.

getURL(url, target, [method])

Load the given URL into the named target. The target argument corresponds to HTML document targets (such as "_top" or "_blank"). The optional method argument can be POST or GET if you want to submit variables back to the server.

loadMovie(url, target)

Load the given URL into the named target. The target argument can be a frame name (I think), or one of the magical values "_level0" (replaces current movie) or "_level1" (loads new movie on top of current movie).

nextFrame()

Go to the next frame.

prevFrame()

Go to the last (or, rather, previous) frame.

play()

Start playing the movie.

stop()

Stop playing the movie.

toggleQuality()

Toggle between high and low quality.

stopSounds()

Stop playing all sounds.

gotoFrame(num)

Go to frame number num. Frame numbers start at 0.

gotoFrame(name)

Go to the frame named name. Which does a lot of good, since I haven't added frame labels yet.

setTarget(expr)

Sets the context for action. Or so they say- I really have no idea what this does.

And there's one weird extra thing. The expression frameLoaded(num) can be used in if statements and while loops to check if the given frame number has been loaded yet. Well, it's supposed to, anyway, but I've never tested it and I seriously doubt it actually works. You can just use /: framesLoaded instead.

Movie clips (all together now- aka sprites) have properties. You can read all of them (or can you?), you can set some of them, and here they are:

- x
- y
- xScale
- yScale
- currentFrame - (read-only)
- totalFrames - (read-only)
- alpha - transparency level
- visible - 1=on, 0=off (?)
- width - (read-only)
- height - (read-only)
- rotation
- target - (read-only) (???)
- framesLoaded - (read-only)
- name
- dropTarget - (read-only) (???)
- url - (read-only) (???)
- highQuality - 1=high, 0=low (?)
- focusRect - (???)

- soundBufTime - (???)

So, setting a sprite's x position is as simple as */box.x = 100;*. Why the slash in front of the box, though? That's how flash keeps track of the sprites in the movie, just like a Unix filesystem- here it shows that box is at the top level. If the sprite named box had another sprite named biff inside of it, you'd set its x position with /box/biff.x = 100;. At least, I think so; correct me if I'm wrong here.

This simple example will move the red square across the window.

Ejemplo 1. swfaction() example

```
<?php
  $s = new SWFShape();
  $f = $s->addFill(0xff, 0, 0);
  $s->setRightFill($f);

  $s->movePenTo(-500, -500);
  $s->drawLineTo(500, -500);
  $s->drawLineTo(500, 500);
  $s->drawLineTo(-500, 500);
  $s->drawLineTo(-500, -500);

  $p = new SWFSprite();
  $i = $p->add($s);
  $i->setDepth(1);
  $p->nextFrame();

  for ($n=0; $n<5; ++$n) {
    $i->rotate(-15);
    $p->nextFrame();
  }

  $m = new SWFMovie();
  $m->setBackground(0xff, 0xff, 0xff);
  $m->setDimension(6000, 4000);

  $i = $m->add($p);
  $i->setDepth(1);
  $i->moveTo(-500,2000);
  $i->setName("box");

  $m->add(new SWFAction("/box.x += 3;"));
  $m->nextFrame();
  $m->add(new SWFAction("gotoFrame(0); play();"));
  $m->nextFrame();

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

This simple example tracks down your mouse on the screen.

Ejemplo 2. swfaction() example

```
<?php

  $m = new SWFMovie();
  $m->setRate(36.0);
  $m->setDimension(1200, 800);
  $m->setBackground(0, 0, 0);

  /* mouse tracking sprite - empty, but follows mouse so we can
     get its x and y coordinates */

  $i = $m->add(new SWFSprite());
  $i->setName('mouse');

  $m->add(new SWFAction("
    startDrag('/mouse', 1); /* '1' means lock sprite to the mouse */
  "));

  /* might as well turn off antialiasing, since these are just squares. */

  $m->add(new SWFAction("
    this.quality = 0;
  "));

  /* morphing box */
  $r = new SWFMorph();
  $s = $r->getShape1();

  /* Note this is backwards from normal shapes.  No idea why. */
  $s->setLeftFill($s->addFill(0xff, 0xff, 0xff));
  $s->movePenTo(-40, -40);
  $s->drawLine(80, 0);
  $s->drawLine(0, 80);
  $s->drawLine(-80, 0);
  $s->drawLine(0, -80);

  $s = $r->getShape2();

  $s->setLeftFill($s->addFill(0x00, 0x00, 0x00));
  $s->movePenTo(-1, -1);
  $s->drawLine(2, 0);
  $s->drawLine(0, 2);
  $s->drawLine(-2, 0);
  $s->drawLine(0, -2);

  /* sprite container for morphing box -
     this is just a timeline w/ the box morphing */

  $box = new SWFSprite();
  $box->add(new SWFAction("
    stop();
  "));
  $i = $box->add($r);

  for ($n=0; $n<=20; ++$n) {
    $i->setRatio($n/20);
    $box->nextFrame();
  }

  /* this container sprite allows us to use the same action code many times */

  $cell = new SWFSprite();
  $i = $cell->add($box);
  $i->setName('box');

  $cell->add(new SWFAction("

    setTarget('box');

    /* ...x means the x coordinate of the parent, i.e. (..).x */
    dx = (/mouse.x + random(6)-3 - ...x)/5;
    dy = (/mouse.y + random(6)-3 - ...y)/5;
    gotoFrame(int(dx*dx + dy*dy));
```

Same as above, but with nice colored balls...

Ejemplo 3. swfaction() example

```
<?php

  $m = new SWFMovie();
  $m->setDimension(11000, 8000);
  $m->setBackground(0x00, 0x00, 0x00);

  $m->add(new SWFAction("

this.quality = 0;
/frames.visible = 0;
startDrag('/mouse', 1);

  "));

  // mouse tracking sprite
  $t = new SWFSprite();
  $i = $m->add($t);
  $i->setName('mouse');

  $g = new SWFGradient();
  $g->addEntry(0, 0xff, 0xff, 0xff, 0xff);
  $g->addEntry(0.1, 0xff, 0xff, 0xff, 0xff);
  $g->addEntry(0.5, 0xff, 0xff, 0xff, 0x5f);
  $g->addEntry(1.0, 0xff, 0xff, 0xff, 0);

  // gradient shape thing
  $s = new SWFShape();
  $f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
  $f->scaleTo(0.03);
  $s->setRightFill($f);
  $s->movePenTo(-600, -600);
  $s->drawLine(1200, 0);
  $s->drawLine(0, 1200);
  $s->drawLine(-1200, 0);
  $s->drawLine(0, -1200);

  // need to make this a sprite so we can multColor it
  $p = new SWFSprite();
  $p->add($s);
  $p->nextFrame();

  // put the shape in here, each frame a different color
  $q = new SWFSprite();
  $q->add(new SWFAction("gotoFrame(random(7)+1); stop();"));
  $i = $q->add($p);

  $i->multColor(1.0, 1.0, 1.0);
  $q->nextFrame();
  $i->multColor(1.0, 0.5, 0.5);
  $q->nextFrame();
  $i->multColor(1.0, 0.75, 0.5);
  $q->nextFrame();
  $i->multColor(1.0, 1.0, 0.5);
  $q->nextFrame();
  $i->multColor(0.5, 1.0, 0.5);
  $q->nextFrame();
  $i->multColor(0.5, 0.5, 1.0);
  $q->nextFrame();
  $i->multColor(1.0, 0.5, 1.0);
  $q->nextFrame();

  // finally, this one contains the action code
  $p = new SWFSprite();
  $i = $p->add($q);
  $i->setName('frames');
  $p->add(new SWFAction("

dx = (/:mousex-/:lastx)/3 + random(10)-5;
dy = (/:mousey-/:lasty)/3;
x = /:mousex;
y = /:mousey;
alpha = 100;
```

SWFBitmap->getHeight

(no version information, might be only in CVS)

SWFBitmap->getHeight -- Returns the bitmap's height

Description

int **swfbitmap->getheight** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfbitmap->getheight() returns the bitmap's height in pixels.

See also **swfbitmap->getwidth()**.

SWFBitmap->getWidth

(no version information, might be only in CVS)

SWFBitmap->getWidth -- Returns the bitmap's width

Description

int **swfbitmap->getwidth** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfbitmap->getwidth() returns the bitmap's width in pixels.

See also **swfbitmap->getheight()**.

SWFBitmap

(PHP 4 >= 4.0.5)

SWFBitmap -- Loads Bitmap object

Description

SWFBitmap **swfbitmap** (mixed file [, mixed alphafile])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfbitmap() creates a new SWFBitmap object from the Jpeg or DBL file in *file*. *alphafile* is a MSK file to be used as an alpha mask for a Jpeg image. Both parameters can be **fopen()** resources or binary strings.

> **Nota:** We can only deal with baseline (frame 0) jpegs, no baseline optimized or progressive scan jpegs!

SWFBitmap has the following methods : **swfbitmap->getwidth()** and **swfbitmap->getheight()**.

You can't import png images directly, though- have to use the png2dbl utility to make a dbl ("define bits lossless") file from the png. The reason for this is that I don't want a dependency on the png library in ming- autoconf should solve this, but that's not set up yet.

Ejemplo 1. Import PNG files

```
<?php
  $s = new SWFShape();
  $f = $s->addFill(new SWFBitmap(file_get_contents("png.dbl")));
  $s->setRightFill($f);

  $s->drawLine(32, 0);
  $s->drawLine(0, 32);
  $s->drawLine(-32, 0);
  $s->drawLine(0, -32);

  $m = new SWFMovie();
  $m->setDimension(32, 32);
  $m->add($s);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

And you can put an alpha mask on a jpeg fill.

Ejemplo 2. swfbitmap() example

```
<?php

  $s = new SWFShape();

  // .msk file generated with "gif2mask" utility
  $f = $s->addFill(new SWFBitmap(file_get_contents("alphafill.jpg"), file_get_contents(
  $s->setRightFill($f);

  $s->drawLine(640, 0);
  $s->drawLine(0, 480);
  $s->drawLine(-640, 0);
  $s->drawLine(0, -480);

  $c = new SWFShape();
  $c->setRightFill($c->addFill(0x99, 0x99, 0x99));
  $c->drawLine(40, 0);
  $c->drawLine(0, 40);
  $c->drawLine(-40, 0);
  $c->drawLine(0, -40);

  $m = new SWFMovie();
  $m->setDimension(640, 480);
  $m->setBackground(0xcc, 0xcc, 0xcc);

  // draw checkerboard background
  for ($y=0; $y<480; $y+=40) {
    for ($x=0; $x<640; $x+=80) {
      $i = $m->add($c);
      $i->moveTo($x, $y);
    }

    $y+=40;

    for ($x=40; $x<640; $x+=80) {
      $i = $m->add($c);
      $i->moveTo($x, $y);
    }
  }

  $m->add($s);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

swfbutton_keypress

(PHP 4 >= 4.0.5)

swfbutton_keypress -- Returns the action flag for keyPress(char)

Description

int **swfbutton_keypress** (string str)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

SWFbutton->addAction

(no version information, might be only in CVS)

SWFbutton->addAction -- Adds an action

Description

void **swfbutton->addaction** (resource action, int flags)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfbutton->addaction() adds the action *action* to this button for the given conditions. The following *flags* are valid: SWFBUTTON_MOUSEOVER, SWFBUTTON_MOUSEOUT, SWFBUTTON_MOUSEUP, SWFBUTTON_MOUSEUPOUTSIDE, SWFBUTTON_MOUSEDOWN, SWFBUTTON_DRAGOUT and SWFBUTTON_DRAGOVER.

See also **swfbutton->addshape()** and **swfaction()**.

SWFbutton->addShape

(no version information, might be only in CVS)

SWFbutton->addShape -- Adds a shape to a button

Description

void **swfbutton->addshape** (resource shape, int flags)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfbutton->addshape() adds the shape *shape* to this button. The following *flags*' values are valid: SWFBUTTON_UP, SWFBUTTON_OVER, SWFBUTTON_DOWN or SWFBUTTON_HIT. SWFBUTTON_HIT isn't ever displayed, it defines the hit region for the button. That is, everywhere the hit shape would be drawn is considered a "touchable" part of the button.

SWFbutton->setAction

(no version information, might be only in CVS)

SWFbutton->setAction -- Sets the action

Description

void **swfbutton->setaction** (resource action)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfbutton->setaction() sets the action to be performed when the button is clicked. Alias for addAction(shape, SWFBUTTON_MOUSEUP). *action* is a **swfaction()**.

See also **swfbutton->addshape()** and **swfaction()**.

SWFbutton->setdown

(no version information, might be only in CVS)

SWFbutton->setdown -- Alias for addShape(shape, SWFBUTTON_DOWN)

Description

void **swfbutton->setdown** (resource shape)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfbutton->setdown() alias for addShape(shape, SWFBUTTON_DOWN).

See also **swfbutton->addshape()** and **swfaction()**.

SWFbutton->setHit

(no version information, might be only in CVS)

SWFbutton->setHit -- Alias for addShape(shape, SWFBUTTON_HIT)

Description

void **swfbutton->sethit** (resource shape)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfbutton->sethit() alias for addShape(shape, SWFBUTTON_HIT).

See also **swfbutton->addshape()** and **swfaction()**.

SWFbutton->setOver

(no version information, might be only in CVS)

SWFbutton->setOver -- Alias for addShape(shape, SWFBUTTON_OVER)

Description

void **swfbutton->setover** (resource shape)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfbutton->setover() alias for addShape(shape, SWFBUTTON_OVER).

See also **swfbutton->addshape()** and **swfaction()**.

SWFbutton->setUp

(no version information, might be only in CVS)

SWFbutton->setUp -- Alias for addShape(shape, SWFBUTTON_UP)

Description

void **swfbutton->setup** (resource shape)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfbutton->setup() alias for addShape(shape, SWFBUTTON_UP).

See also **swfbutton->addshape()** and **swfaction()**.

SWFbutton

(PHP 4 >= 4.0.5)

SWFbutton -- Creates a new Button

Description

SWFButton **swfbutton** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfbutton() creates a new Button. Roll over it, click it, see it call action code. Swank.

SWFButton has the following methods : **swfbutton->addshape()**, **swfbutton->setup()**, **swfbutton->setover() swfbutton->setdown()**, **swfbutton->sethit() swfbutton->setaction()** and **swfbutton->addaction()**.

This simple example will show your usual interactions with buttons : rollover, rollon, mouseup, mousedown, noaction.
Ejemplo 1. swfbutton() example

```
<?php

  $f = new SWFFont("_serif");

  $p = new SWFSprite();

  function label($string)
  {
    global $f;

    $t = new SWFTextField();
    $t->setFont($f);
    $t->addString($string);
    $t->setHeight(200);
    $t->setBounds(3200, 200);
    return $t;
  }

  function addLabel($string)
  {
    global $p;

    $i = $p->add(label($string));
    $p->nextFrame();
    $p->remove($i);
  }

  $p->add(new SWFAction("stop();"));
  addLabel("NO ACTION");
  addLabel("SWFBUTTON_MOUSEUP");
  addLabel("SWFBUTTON_MOUSEDOWN");
  addLabel("SWFBUTTON_MOUSEOVER");
  addLabel("SWFBUTTON_MOUSEOUT");
  addLabel("SWFBUTTON_MOUSEUPOUTSIDE");
  addLabel("SWFBUTTON_DRAGOVER");
  addLabel("SWFBUTTON_DRAGOUT");

  function rect($r, $g, $b)
  {
    $s = new SWFShape();
    $s->setRightFill($s->addFill($r, $g, $b));
    $s->drawLine(600, 0);
    $s->drawLine(0, 600);
    $s->drawLine(-600, 0);
    $s->drawLine(0, -600);

    return $s;
  }

  $b = new SWFButton();
  $b->addShape(rect(0xff, 0, 0), SWFBUTTON_UP | SWFBUTTON_HIT);
  $b->addShape(rect(0, 0xff, 0), SWFBUTTON_OVER);
  $b->addShape(rect(0, 0, 0xff), SWFBUTTON_DOWN);

  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(1);"),
            SWFBUTTON_MOUSEUP);

  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(2);"),
        SWFBUTTON_MOUSEDOWN);

  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(3);"),
        SWFBUTTON_MOUSEOVER);

  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(4);"),
        SWFBUTTON_MOUSEOUT);

  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(5);"),
        SWFBUTTON_MOUSEUPOUTSIDE);

  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(6);"),
        SWFBUTTON_DRAGOVER);
```

This simple example will enables you to drag draw a big red button on the windows. No drag-and-drop, just moving around.

Ejemplo 2. swfbutton->addaction() example

```
<?php

  $s = new SWFShape();
  $s->setRightFill($s->addFill(0xff, 0, 0));
  $s->drawLine(1000,0);
  $s->drawLine(0,1000);
  $s->drawLine(-1000,0);
  $s->drawLine(0,-1000);

  $b = new SWFButton();
  $b->addShape($s, SWFBUTTON_HIT | SWFBUTTON_UP | SWFBUTTON_DOWN | SWFBUTTON_OVER);

  $b->addAction(new SWFAction("startDrag('/test', 0);"), // '0' means don't lock to mou
        SWFBUTTON_MOUSEDOWN);

  $b->addAction(new SWFAction("stopDrag();"),
        SWFBUTTON_MOUSEUP | SWFBUTTON_MOUSEUPOUTSIDE);

  $p = new SWFSprite();
  $p->add($b);
  $p->nextFrame();

  $m = new SWFMovie();
  $i = $m->add($p);
  $i->setName('test');
  $i->moveTo(1000,1000);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFDisplayItem->addColor

(no version information, might be only in CVS)

SWFDisplayItem->addColor -- Adds the given color to this item's color transform

Description

void **swfdisplayitem->addcolor** ([int red [, int green [, int blue [, int a]]]])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfdisplayitem->addcolor() adds the color to this item's color transform. The color is given in its RGB form.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

SWFDisplayItem->move

(no version information, might be only in CVS)

SWFDisplayItem->move -- Moves object in relative coordinates

Description

void **swfdisplayitem->move** (int dx, int dy)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfdisplayitem->move() moves the current object by (*dx*,*dy*) from its current position.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->moveto()**.

SWFDisplayItem->moveTo

(no version information, might be only in CVS)

SWFDisplayItem->moveTo -- Moves object in global coordinates

Description

void **swfdisplayitem->moveto** (int x, int y)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfdisplayitem->moveto() moves the current object to (*x*,*y*) in global coordinates.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->move()**.

SWFDisplayItem->multColor

(no version information, might be only in CVS)

SWFDisplayItem->multColor -- Multiplies the item's color transform

Description

void **swfdisplayitem->multcolor** ([int red [, int green [, int blue [, int a]]]])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfdisplayitem->multcolor() multiplies the item's color transform by the given values.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

This simple example will modify your picture's atmosphere to Halloween (use a landscape or bright picture).

Ejemplo 1. swfdisplayitem->multcolor() example

```
<?php

  $b = new SWFBitmap(file_get_contents("backyard.jpg"));
  // note use your own picture :-)
  $s = new SWFShape();
  $s->setRightFill($s->addFill($b));
  $s->drawLine($b->getWidth(), 0);
  $s->drawLine(0, $b->getHeight());
  $s->drawLine(-$b->getWidth(), 0);
  $s->drawLine(0, -$b->getHeight());

  $m = new SWFMovie();
  $m->setDimension($b->getWidth(), $b->getHeight());

  $i = $m->add($s);

  for ($n=0; $n<=20; ++$n) {
    $i->multColor(1.0-$n/10, 1.0, 1.0);
    $i->addColor(0xff*$n/20, 0, 0);
    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFDisplayItem->remove

(no version information, might be only in CVS)

SWFDisplayItem->remove -- Removes the object from the movie

Description

void **swfdisplayitem->remove** (void)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfdisplayitem->remove() removes this object from the movie's display list.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

See also **swfmovie->add()**.

SWFDisplayItem->Rotate

(no version information, might be only in CVS)

SWFDisplayItem->Rotate -- Rotates in relative coordinates

Description

void **swfdisplayitem->rotate** (float ddegrees)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfdisplayitem->rotate() rotates the current object by *ddegrees* degrees from its current rotation.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->rotateto()**.

SWFDisplayItem->rotateTo

(no version information, might be only in CVS)

SWFDisplayItem->rotateTo -- Rotates the object in global coordinates

Description

void **swfdisplayitem->rotateto** (float degrees)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfdisplayitem->rotateto() set the current object rotation to *degrees* degrees in global coordinates.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

This example bring three rotating string from the background to the foreground. Pretty nice.
Ejemplo 1. swfdisplayitem->rotateto() example

```
<?php
  $thetext =  "ming!";

  $f = new SWFFont("Bauhaus 93.fdb");

  $m = new SWFMovie();
  $m->setRate(24.0);
  $m->setDimension(2400, 1600);
  $m->setBackground(0xff, 0xff, 0xff);

  // functions with huge numbers of arbitrary
  // arguments are always a good idea!  Really!

  function text($r, $g, $b, $a, $rot, $x, $y, $scale, $string)
  {
    global $f, $m;

    $t = new SWFText();
    $t->setFont($f);
    $t->setColor($r, $g, $b, $a);
    $t->setHeight(960);
    $t->moveTo(-($f->getWidth($string))/2, $f->getAscent()/2);
    $t->addString($string);

    // we can add properties just like a normal PHP var,
    // as long as the names aren't already used.
    // e.g., we can't set $i->scale, because that's a function

    $i = $m->add($t);
    $i->x = $x;
    $i->y = $y;
    $i->rot = $rot;
    $i->s = $scale;
    $i->rotateTo($rot);
    $i->scale($scale, $scale);

    // but the changes are local to the function, so we have to
    // return the changed object.  kinda weird..

    return $i;
 }

  function step($i)
  {
    $oldrot = $i->rot;
    $i->rot = 19*$i->rot/20;
    $i->x = (19*$i->x + 1200)/20;
    $i->y = (19*$i->y + 800)/20;
    $i->s = (19*$i->s + 1.0)/20;

    $i->rotateTo($i->rot);
    $i->scaleTo($i->s, $i->s);
    $i->moveTo($i->x, $i->y);

    return $i;
  }

  // see?  it sure paid off in legibility:

  $i1 = text(0xff, 0x33, 0x33, 0xff, 900, 1200, 800, 0.03, $thetext);
  $i2 = text(0x00, 0x33, 0xff, 0x7f, -560, 1200, 800, 0.04, $thetext);
  $i3 = text(0xff, 0xff, 0xff, 0x9f, 180, 1200, 800, 0.001, $thetext);

  for ($i=1; $i<=100; ++$i) {
    $i1 = step($i1);
    $i2 = step($i2);
    $i3 = step($i3);

    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
```

See also **swfdisplayitem->rotate()**.

SWFDisplayItem->scale

(no version information, might be only in CVS)

SWFDisplayItem->scale -- Scales the object in relative coordinates

Description

void **swfdisplayitem->scale** (int dx, int dy)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfdisplayitem->scale() scales the current object by (*dx*,*dy*) from its current size.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->scaleto()**.

SWFDisplayItem->scaleTo

(no version information, might be only in CVS)

SWFDisplayItem->scaleTo -- Scales the object in global coordinates

Description

void **swfdisplayitem->scaleto** (int x, int y)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfdisplayitem->scaleto() scales the current object to (*x*,*y*) in global coordinates.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->scale()**.

SWFDisplayItem->setDepth

(no version information, might be only in CVS)

SWFDisplayItem->setDepth -- Sets z-order

Description

void **swfdisplayitem->setdepth** (float depth)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfdisplayitem->setdepth() sets the object's z-order to *depth*. Depth defaults to the order in which instances are created (by adding a shape/text to a movie)- newer ones are on top of older ones. If two objects are given the same depth, only the later-defined one can be moved.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

SWFDisplayItem->setName

(no version information, might be only in CVS)

SWFDisplayItem->setName -- Sets the object's name

Description

void **swfdisplayitem->setname** (string name)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfdisplayitem->setname() sets the object's name to *name*, for targetting with action script. Only useful on sprites.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

SWFDisplayItem->setRatio

(no version information, might be only in CVS)

SWFDisplayItem->setRatio -- Sets the object's ratio

Description

void **swfdisplayitem->setratio** (float ratio)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfdisplayitem->setratio() sets the object's ratio to *ratio*. Obviously only useful for morphs.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

This simple example will morph nicely three concentric circles.
Ejemplo 1. swfdisplayitem->setname() example

```
<?php

  $p = new SWFMorph();

  $g = new SWFGradient();
  $g->addEntry(0.0, 0, 0, 0);
  $g->addEntry(0.16, 0xff, 0xff, 0xff);
  $g->addEntry(0.32, 0, 0, 0);
  $g->addEntry(0.48, 0xff, 0xff, 0xff);
  $g->addEntry(0.64, 0, 0, 0);
  $g->addEntry(0.80, 0xff, 0xff, 0xff);
  $g->addEntry(1.00, 0, 0, 0);

  $s = $p->getShape1();
  $f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
  $f->scaleTo(0.05);
  $s->setLeftFill($f);
  $s->movePenTo(-160, -120);
  $s->drawLine(320, 0);
  $s->drawLine(0, 240);
  $s->drawLine(-320, 0);
  $s->drawLine(0, -240);

  $g = new SWFGradient();
  $g->addEntry(0.0, 0, 0, 0);
  $g->addEntry(0.16, 0xff, 0, 0);
  $g->addEntry(0.32, 0, 0, 0);
  $g->addEntry(0.48, 0, 0xff, 0);
  $g->addEntry(0.64, 0, 0, 0);
  $g->addEntry(0.80, 0, 0, 0xff);
  $g->addEntry(1.00, 0, 0, 0);

  $s = $p->getShape2();
  $f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
  $f->scaleTo(0.05);
  $f->skewXTo(1.0);
  $s->setLeftFill($f);
  $s->movePenTo(-160, -120);
  $s->drawLine(320, 0);
  $s->drawLine(0, 240);
  $s->drawLine(-320, 0);
  $s->drawLine(0, -240);

  $m = new SWFMovie();
  $m->setDimension(320, 240);
  $i = $m->add($p);
  $i->moveTo(160, 120);

  for ($n=0; $n<=1.001; $n+=0.01) {
    $i->setRatio($n);
    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFDisplayItem->skewX

(no version information, might be only in CVS)

SWFDisplayItem->skewX -- Sets the X-skew

Description

void **swfdisplayitem->skewx** (float ddegrees)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfdisplayitem->skewx() adds *ddegrees* to current x-skew.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->skewx()**, **swfdisplayitem->skewy()** and **swfdisplayitem->skewyto()**.

SWFDisplayItem->skewXTo

(no version information, might be only in CVS)

SWFDisplayItem->skewXTo -- Sets the X-skew

Description

void **swfdisplayitem->skewxto** (float degrees)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfdisplayitem->skewxto() sets the x-skew to *degrees*. For *degrees* is 1.0, it means a 45-degree forward slant. More is more forward, less is more backward.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->skewx()**, **swfdisplayitem->skewy()** and **swfdisplayitem->skewyto()**.

SWFDisplayItem->skewY

(no version information, might be only in CVS)

SWFDisplayItem->skewY -- Sets the Y-skew

Description

void **swfdisplayitem->skewy** (float ddegrees)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfdisplayitem->skewy() adds *ddegrees* to current y-skew.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->skewyto()**, **swfdisplayitem->skewx()** and **swfdisplayitem->skewxto()**.

SWFDisplayItem->skewYTo

(no version information, might be only in CVS)

SWFDisplayItem->skewYTo -- Sets the Y-skew

Description

void **swfdisplayitem->skewyto** (float degrees)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfdisplayitem->skewyto() sets the y-skew to *degrees*. For *degrees* is 1.0, it means a 45-degree forward slant. More is more upward, less is more downward.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->skewy()**, **swfdisplayitem->skewx()** and **swfdisplayitem->skewxto()**.

SWFDisplayItem

(no version information, might be only in CVS)

SWFDisplayItem -- Creates a new displayitem object

Description

SWFDisplayItem **swfdisplayitem** (void)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfdisplayitem() creates a new swfdisplayitem object.

Here's where all the animation takes place. After you define a shape, a text object, a sprite, or a button, you add it to the movie, then use the returned handle to move, rotate, scale, or skew the thing.

SWFDisplayItem has the following methods : **swfdisplayitem->move()**, **swfdisplayitem->moveto ()**, **swfdisplayitem->scaleto()**, **swfdisplayitem->scale()**, **swfdisplayitem->rotate()**, **swfdisplayitem->rotateto()**, **swfdisplayitem->skewxto()**, **swfdisplayitem->skewx()**, **swfdisplayitem->skewyto() swfdisplayitem->skewyto()**, **swfdisplayitem->setdepth() swfdisplayitem->remove()**, **swfdisplayitem->setname() swfdisplayitem->setratio()**, **swfdisplayitem->addcolor()** and **swfdisplayitem->multcolor()**.

SWFFill->moveTo

(no version information, might be only in CVS)

SWFFill->moveTo -- Moves fill origin

Description

void **swffill->moveto** (int x, int y)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swffill->moveto() moves fill's origin to (*x*,*y*) in global coordinates.

SWFFill->rotateTo

(no version information, might be only in CVS)

SWFFill->rotateTo -- Sets fill's rotation

Description

void **swffill->rotateto** (float degrees)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swffill->rotateto() sets fill's rotation to *degrees* degrees.

SWFFill->scaleTo

(no version information, might be only in CVS)

SWFFill->scaleTo -- Sets fill's scale

Description

void **swffill->scaleto** (int x, int y)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swffill->scaleto() sets fill's scale to *x* in the x-direction, *y* in the y-direction.

SWFFill->skewXTo

(no version information, might be only in CVS)

SWFFill->skewXTo -- Sets fill x-skew

Description

void **swffill->skewxto** (float x)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swffill->skewxto() sets fill x-skew to *x*. For *x* is 1.0, it is a 45-degree forward slant. More is more forward, less is more backward.

SWFFill->skewYTo

(no version information, might be only in CVS)

SWFFill->skewYTo -- Sets fill y-skew

Description

void **swffill->skewyto** (float y)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swffill->skewyto() sets fill y-skew to *y*. For *y* is 1.0, it is a 45-degree upward slant. More is more upward, less is more downward.

SWFFill

(PHP 4 >= 4.0.5)

SWFFill -- Loads SWFFill object

Description

SWFFill **swffill** (void)

The **swffill()** object allows you to transform (scale, skew, rotate) bitmap and gradient fills. **swffill()** objects are created by the **swfshape->addfill()** methods.

SWFFill has the following methods: **swffill->moveto()** and **swffill->scaleto()**, **swffill->rotateto()**, **swffill->skewxto()** and **swffill->skewyto()**.

swffont->getwidth

(no version information, might be only in CVS)

swffont->getwidth -- Returns the string's width

Description

int **swffont->getwidth** (string string)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swffont->getwidth() returns the string *string*'s width, using font's default scaling. You'll probably want to use the **swftext()** version of this method which uses the text object's scale.

SWFFont

(PHP 4 >= 4.0.5)

SWFFont -- Loads a font definition

Description

SWFFont **swffont** (string filename)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

If *filename* is the name of an FDB file (i.e., it ends in ".fdb"), load the font definition found in said file. Otherwise, create a browser-defined font reference.

FDB ("font definition block") is a very simple wrapper for the SWF DefineFont2 block which contains a full description of a font. One may create FDB files from SWT Generator template files with the included makefdb utility- look in the util directory off the main ming distribution directory.

Browser-defined fonts don't contain any information about the font other than its name. It is assumed that the font definition will be provided by the movie player. The fonts _serif, _sans, and _typewriter should always be available. For example:

```
<?php
$f = newSWFFont("_sans");
?>
```

will give you the standard sans-serif font, probably the same as what you'd get with <*font name="sans-serif"*> in HTML.

swffont() returns a reference to the font definition, for use in the **swftext->setfont()** and the **swftextfield->setfont()** methods.

SWFFont has the following methods : **swffont->getwidth()**.

SWFGradient->addEntry

(no version information, might be only in CVS)

SWFGradient->addEntry -- Adds an entry to the gradient list

Description

void **swfgradient->addentry** (float ratio, int red, int green, int blue [, int a])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfgradient->addentry() adds an entry to the gradient list. *ratio* is a number between 0 and 1 indicating where in the gradient this color appears. Thou shalt add entries in order of increasing ratio.

red, *green*, *blue* is a color (RGB mode). Last parameter *a* is optional.

SWFGradient

(PHP 4 >= 4.0.5)

SWFGradient -- Creates a gradient object

Description

SWFGradient **swfgradient** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfgradient() creates a new SWFGradient object.

After you've added the entries to your gradient, you can use the gradient in a shape fill with the **swfshape->addfill()** method.

SWFGradient has the following methods : **swfgradient->addentry()**.

This simple example will draw a big black-to-white gradient as background, and a reddish disc in its center.

Ejemplo 1. swfgradient() example

```
<?php

  $m = new SWFMovie();
  $m->setDimension(320, 240);

  $s = new SWFShape();

  // first gradient- black to white
  $g = new SWFGradient();
  $g->addEntry(0.0, 0, 0, 0);
  $g->addEntry(1.0, 0xff, 0xff, 0xff);

  $f = $s->addFill($g, SWFFILL_LINEAR_GRADIENT);
  $f->scaleTo(0.01);
  $f->moveTo(160, 120);
  $s->setRightFill($f);
  $s->drawLine(320, 0);
  $s->drawLine(0, 240);
  $s->drawLine(-320, 0);
  $s->drawLine(0, -240);

  $m->add($s);

  $s = new SWFShape();

  // second gradient- radial gradient from red to transparent
  $g = new SWFGradient();
  $g->addEntry(0.0, 0xff, 0, 0, 0xff);
  $g->addEntry(1.0, 0xff, 0, 0, 0);

  $f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
  $f->scaleTo(0.005);
  $f->moveTo(160, 120);
  $s->setRightFill($f);
  $s->drawLine(320, 0);
  $s->drawLine(0, 240);
  $s->drawLine(-320, 0);
  $s->drawLine(0, -240);

  $m->add($s);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFMorph->getshape1

(no version information, might be only in CVS)

SWFMorph->getshape1 -- Gets a handle to the starting shape

Description

mixed **swfmorph->getshape1** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfmorph->getshape1() gets a handle to the morph's starting shape. **swfmorph->getshape1()** returns an **swfshape()** object.

SWFMorph->getshape2

(no version information, might be only in CVS)

SWFMorph->getshape2 -- Gets a handle to the ending shape

Description

mixed **swfmorph->getshape2** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfmorph->getshape2() gets a handle to the morph's ending shape. **swfmorph->getshape2()** returns an **swfshape()** object.

SWFMorph

(PHP 4 >= 4.0.5)

SWFMorph -- Creates a new SWFMorph object

Description

SWFMorph **swfmorph** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfmorph() creates a new SWFMorph object.

Also called a "shape tween". This thing lets you make those tacky twisting things that make your computer choke. Oh, joy!

The methods here are sort of weird. It would make more sense to just have newSWFMorph(shape1, shape2);, but as things are now, shape2 needs to know that it's the second part of a morph. (This, because it starts writing its output as soon as it gets drawing commands- if it kept its own description of its shapes and wrote on completion this and some other things would be much easier.)

SWFMorph has the following methods : **swfmorph->getshape1()** and **swfmorph->getshape1()**.

This simple example will morph a big red square into a smaller blue black-bordered square.
Ejemplo 1. swfmorph() example

```
<?php
  $p = new SWFMorph();

  $s = $p->getShape1();
  $s->setLine(0, 0, 0, 0);

  /* Note that this is backwards from normal shapes (left instead of right).
     I have no idea why, but this seems to work.. */

  $s->setLeftFill($s->addFill(0xff, 0, 0));
  $s->movePenTo(-1000,-1000);
  $s->drawLine(2000,0);
  $s->drawLine(0,2000);
  $s->drawLine(-2000,0);
  $s->drawLine(0,-2000);

  $s = $p->getShape2();
  $s->setLine(60,0,0,0);
  $s->setLeftFill($s->addFill(0, 0, 0xff));
  $s->movePenTo(0,-1000);
  $s->drawLine(1000,1000);
  $s->drawLine(-1000,1000);
  $s->drawLine(-1000,-1000);
  $s->drawLine(1000,-1000);

  $m = new SWFMovie();
  $m->setDimension(3000,2000);
  $m->setBackground(0xff, 0xff, 0xff);

  $i = $m->add($p);
  $i->moveTo(1500,1000);

  for ($r=0.0; $r<=1.0; $r+=0.1) {
    $i->setRatio($r);
    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFMovie->add

(no version information, might be only in CVS)

SWFMovie->add -- Adds any type of data to a movie

Description

void **swfmovie->add** (resource instance)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfmovie->add() adds *instance* to the current movie. *instance* is any type of data : Shapes, text, fonts, etc. must all be added to the movie to make this work.

For displayable types (shape, text, button, sprite), this returns an **swfdisplayitem()**, a handle to the object in a display list. Thus, you can add the same shape to a movie multiple times and get separate

handles back for each separate instance.

See also all other objects (adding this later), and **swfmovie->remove()**

See examples in : **swfdisplayitem->rotateto()** and **swfshape->addfill()**.

SWFMovie->nextframe

(no version information, might be only in CVS)

SWFMovie->nextframe -- Moves to the next frame of the animation

Description

void **swfmovie->nextframe** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfmovie->nextframe() moves to the next frame of the animation.

SWFMovie->output

(no version information, might be only in CVS)

SWFMovie->output -- Dumps your lovingly prepared movie out

Description

int **swfmovie->output** ([int compression])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfmovie->output() dumps your lovingly prepared movie out. In PHP, preceding this with the command

```
<?php
header('Content-type: application/x-shockwave-flash');
?>
```

convinces the browser to display this as a flash movie.

The *compression* level can be a value between 0 and 9, defining the swf compression similar to gzip compression.

See also **swfmovie->save()**.

See examples in : **swfmovie->streammp3()**, **swfdisplayitem->rotateto()**, **swfaction()**... Any example will use this method.

swfmovie->remove

(no version information, might be only in CVS)

swfmovie->remove -- Removes the object instance from the display list

Description

void **swfmovie->remove** (resource instance)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfmovie->remove() removes the object instance *instance* from the display list.

See also **swfmovie->add()**.

SWFMovie->save

(no version information, might be only in CVS)

SWFMovie->save -- Saves your movie in a file

Description

int **swfmovie->save** (string filename [, int compression])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfmovie->save() saves your movie to the file named *filename*.

The *compression* level can be a value between 0 and 9, defining the swf compression similar to gzip compression.

See also **swfmovie->output()**.

SWFMovie->setbackground

(no version information, might be only in CVS)

SWFMovie->setbackground -- Sets the background color

Description

void **swfmovie->setbackground** (int red, int green, int blue)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfmovie->setbackground() sets the background color. Why is there no rgba version? Think about it. (Actually, that's not such a dumb question after all- you might want to let the HTML background show through. There's a way to do that, but it only works on IE4. Search the http://www.macromedia.com/ site for details.)

SWFMovie->setdimension

(no version information, might be only in CVS)

SWFMovie->setdimension -- Sets the movie's width and height

Description

void **swfmovie->setdimension** (int width, int height)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfmovie->setdimension() sets the movie's width to *width* and height to *height*.

SWFMovie->setframes

(no version information, might be only in CVS)

SWFMovie->setframes -- Sets the total number of frames in the animation

Description

void **swfmovie->setframes** (string numberofframes)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfmovie->setframes() sets the total number of frames in the animation to *numberofframes*.

SWFMovie->setrate

(no version information, might be only in CVS)

SWFMovie->setrate -- Sets the animation's frame rate

Description

void **swfmovie->setrate** (int rate)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfmovie->setrate() sets the frame rate to *rate*, in frame per seconds. Animation will slow down if the player can't render frames fast enough- unless there's a streaming sound, in which case display frames are sacrificed to keep sound from skipping.

SWFMovie->streammp3

(no version information, might be only in CVS)

SWFMovie->streammp3 -- Streams a MP3 file

Description

void **swfmovie->streammp3** (mixed mp3File)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfmovie->streammp3() streams the mp3 file *mp3File*. Not very robust in dealing with oddities (can skip over an initial ID3 tag, but that's about it). Like **swfshape->addjpegfill()**, this isn't a stable function- we'll probably need to make a separate SWFSound object to contain sound types. Parameter *mp3File* can be a **fopen()** resource or a binary string.

Note that the movie isn't smart enough to put enough frames in to contain the entire mp3 stream- you'll have to add (length of song * frames per second) frames to get the entire stream in.

Yes, now you can use ming to put that rock and roll devil worship music into your SWF files. Just don't tell the RIAA.

Ejemplo 1. swfmovie->streammp3() example

```
<?php
  $m = new SWFMovie();
  $m->setRate(12.0);
  $m->streamMp3(file_get_contents("distortobass.mp3"));
  // use your own MP3

  // 11.85 seconds at 12.0 fps = 142 frames
  $m->setFrames(142);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFMovie

(PHP 4 >= 4.0.5)

SWFMovie -- Creates a new movie object, representing an SWF version 4 movie

Description

SWFMovie **swfmovie** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfmovie() creates a new movie object, representing an SWF version 4 movie.

SWFMovie has the following methods : **swfmovie->output()**,**swfmovie->save()**, **swfmovie->add()**, **swfmovie->remove()**, **swfmovie->nextframe()**, **swfmovie->setbackground()**, **swfmovie->setrate()**, **swfmovie->setdimension()**, **swfmovie->setframes()** and **swfmovie->streammp3()**.

See examples in : **swfdisplayitem->rotateto()**, **swfshape->setline()**, **swfshape->addfill()**... Any example will use this object.

SWFShape->addFill

(no version information, might be only in CVS)

SWFShape->addFill -- Adds a solid fill to the shape

Description

SWFFill **SWFShape->addFill** (int red, int green, int blue [, int a])

SWFFill **SWFShape->addFill** (SWFBitmap bitmap [, int flags])

SWFFill **SWFShape->addFill** (SWFGradient gradient [, int flags])

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

SWFShape->addFill() adds a solid fill to the shape's list of fill styles. **SWFShape->addFill()** accepts three different types of arguments.

red, *green*, *blue* is a color (RGB mode). Last parameter *a* is optional.

The *bitmap* argument is an **SWFBitmap()** object. The *flags* argument can be one of the following values: SWFFILL_CLIPPED_BITMAP, SWFFILL_TILED_BITMAP, SWFFILL_LINEAR_GRADIENT or SWFFILL_RADIAL_GRADIENT. Default is SWFFILL_TILED_BITMAP for SWFBitmap and SWFFILL_LINEAR_GRADIENT for SWFGradient.

The *gradient* argument is an **SWFGradient()** object. The flags argument can be one of the following values : SWFFILL_RADIAL_GRADIENT or SWFFILL_LINEAR_GRADIENT. Default is SWFFILL_LINEAR_GRADIENT. I'm sure about this one. Really.

SWFShape->addFill() returns an **SWFFill()** object for use with the **SWFShape->setLeftFill()** and **SWFShape->setRightFill()** functions described below.

This simple example will draw a frame on a bitmap. Ah, here's another buglet in the flash player- it doesn't seem to care about the second shape's bitmap's transformation in a morph. According to spec, the bitmap should stretch along with the shape in this example..

Ejemplo 1. SWFShape->addFill() example

```
<?php

  $p = new SWFMorph();

  $b = new SWFBitmap(file_get_contents("alphafill.jpg"));
  // use your own bitmap
  $width = $b->getWidth();
  $height = $b->getHeight();

  $s = $p->getShape1();
  $f = $s->addFill($b, SWFFILL_TILED_BITMAP);
  $f->moveTo(-$width/2, -$height/4);
  $f->scaleTo(1.0, 0.5);
  $s->setLeftFill($f);
  $s->movePenTo(-$width/2, -$height/4);
  $s->drawLine($width, 0);
  $s->drawLine(0, $height/2);
  $s->drawLine(-$width, 0);
  $s->drawLine(0, -$height/2);

  $s = $p->getShape2();
  $f = $s->addFill($b, SWFFILL_TILED_BITMAP);

  // these two have no effect!
  $f->moveTo(-$width/4, -$height/2);
  $f->scaleTo(0.5, 1.0);

  $s->setLeftFill($f);
  $s->movePenTo(-$width/4, -$height/2);
  $s->drawLine($width/2, 0);
  $s->drawLine(0, $height);
  $s->drawLine(-$width/2, 0);
  $s->drawLine(0, -$height);

  $m = new SWFMovie();
  $m->setDimension($width, $height);
  $i = $m->add($p);
  $i->moveTo($width/2, $height/2);

  for ($n=0; $n<1.001; $n+=0.03) {
    $i->setRatio($n);
    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

See also **SWFShape->setLeftFill()** and **SWFShape->setRightFill()**.

SWFShape->drawCurve

(no version information, might be only in CVS)

SWFShape->drawCurve -- Draws a curve (relative)

Description

void **swfshape->drawcurve** (int controldx, int controldy, int anchordx, int anchordy)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfshape->drawcurve() draws a quadratic curve (using the current line style,set by **swfshape->setline()**) from the current pen position to the relative position (*anchorx*,*anchory*) using relative control point (*controlx*,*controly*). That is, head towards the control point, then smoothly turn to the anchor point.

See also **swfshape->drawlineto()**, **swfshape->drawline()**, **swfshape->movepento()** and **swfshape->movepen()**.

SWFShape->drawCurveTo

(no version information, might be only in CVS)

SWFShape->drawCurveTo -- Draws a curve

Description

void **swfshape->drawcurveto** (int controlx, int controly, int anchorx, int anchory)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfshape->drawcurveto() draws a quadratic curve (using the current line style, set by **swfshape->setline()**) from the current pen position to (*anchorx*,*anchory*) using (*controlx*,*controly*) as a control point. That is, head towards the control point, then smoothly turn to the anchor point.

See also **swfshape->drawlineto()**, **swfshape->drawline()**, **swfshape->movepento()** and **swfshape->movepen()**.

SWFShape->drawLine

(no version information, might be only in CVS)

SWFShape->drawLine -- Draws a line (relative)

Description

void **swfshape->drawline** (int dx, int dy)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfshape->drawline() draws a line (using the current line style set by **swfshape->setline()**) from the current pen position to displacement (*dx*,*dy*).

See also **swfshape->movepento()**, **swfshape->drawcurveto()**, **swfshape->movepen()** and **swfshape->drawlineto()**.

SWFShape->drawLineTo

(no version information, might be only in CVS)

SWFShape->drawLineTo -- Draws a line

Description

void **swfshape->drawlineto** (int x, int y)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfshape->setrightfill() draws a line (using the current line style, set by **swfshape->setline()**) from the current pen position to point (*x*,*y*) in the shape's coordinate space.

See also **swfshape->movepento()**, **swfshape->drawcurveto()**, **swfshape->movepen()** and **swfshape->drawline()**.

SWFShape->movePen

(no version information, might be only in CVS)

SWFShape->movePen -- Moves the shape's pen (relative)

Description

void **swfshape->movepen** (int dx, int dy)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfshape->setrightfill() move the shape's pen from coordinates (current x,current y) to (current x + *dx*, current y + *dy*) in the shape's coordinate space.

See also **swfshape->movepento()**, **swfshape->drawcurveto()**, **swfshape->drawlineto()** and **swfshape->drawline()**.

SWFShape->movePenTo

(no version information, might be only in CVS)

SWFShape->movePenTo -- Moves the shape's pen

Description

void **swfshape->movepento** (int x, int y)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfshape->setrightfill() move the shape's pen to (*x*,*y*) in the shape's coordinate space.

See also **swfshape->movepen()**, **swfshape->drawcurveto()**, **swfshape->drawlineto()** and **swfshape->drawline()**.

SWFShape->setLeftFill

(no version information, might be only in CVS)

SWFShape->setLeftFill -- Sets left rasterizing color

Description

void **swfshape->setleftfill** (swfgradient fill)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

void **swfshape->setleftfill** (int red, int green, int blue [, int a])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

What this nonsense is about is, every edge segment borders at most two fills. When rasterizing the object, it's pretty handy to know what those fills are ahead of time, so the swf format requires these to be specified.

swfshape->setleftfill() sets the fill on the left side of the edge- that is, on the interior if you're defining the outline of the shape in a counter-clockwise fashion. The fill object is an SWFFill object

returned from one of the addFill functions above.

This seems to be reversed when you're defining a shape in a morph, though. If your browser crashes, just try setting the fill on the other side.

Shortcut for *swfshape->setleftfill($s->addfill($r, $g, $b [, $a]));*.

See also **swfshape->setrightfill()**.

SWFShape->setLine

(no version information, might be only in CVS)

SWFShape->setLine -- Sets the shape's line style

Description

void **swfshape->setline** (int width [, int red [, int green [, int blue [, int a]]]])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfshape->setline() sets the shape's line style. *width* is the line's width. If *width* is 0, the line's style is removed (then, all other arguments are ignored). If *width* > 0, then line's color is set to *red*, *green*, *blue*. Last parameter *a* is optional.

swfshape->setline() accepts 1, 4 or 5 arguments (not 3 or 2).

You must declare all line styles before you use them (see example).

This simple example will draw a big "!#%*@", in funny colors and gracious style.
Ejemplo 1. swfshape->setline() example

```php
<?php
  $s = new SWFShape();
  $f1 = $s->addFill(0xff, 0, 0);
  $f2 = $s->addFill(0xff, 0x7f, 0);
  $f3 = $s->addFill(0xff, 0xff, 0);
  $f4 = $s->addFill(0, 0xff, 0);
  $f5 = $s->addFill(0, 0, 0xff);

  // bug: have to declare all line styles before you use them
  $s->setLine(40, 0x7f, 0, 0);
  $s->setLine(40, 0x7f, 0x3f, 0);
  $s->setLine(40, 0x7f, 0x7f, 0);
  $s->setLine(40, 0, 0x7f, 0);
  $s->setLine(40, 0, 0, 0x7f);

  $f = new SWFFont('Techno.fdb');

  $s->setRightFill($f1);
  $s->setLine(40, 0x7f, 0, 0);
  $s->drawGlyph($f, '!');
  $s->movePen($f->getWidth('!'), 0);

  $s->setRightFill($f2);
  $s->setLine(40, 0x7f, 0x3f, 0);
  $s->drawGlyph($f, '#');
  $s->movePen($f->getWidth('#'), 0);

  $s->setRightFill($f3);
  $s->setLine(40, 0x7f, 0x7f, 0);
  $s->drawGlyph($f, '%');
  $s->movePen($f->getWidth('%'), 0);

  $s->setRightFill($f4);
  $s->setLine(40, 0, 0x7f, 0);
  $s->drawGlyph($f, '*');
  $s->movePen($f->getWidth('*'), 0);

  $s->setRightFill($f5);
  $s->setLine(40, 0, 0, 0x7f);
  $s->drawGlyph($f, '@');

  $m = new SWFMovie();
  $m->setDimension(3000,2000);
  $m->setRate(12.0);
  $i = $m->add($s);
  $i->moveTo(1500-$f->getWidth("!#%*@")/2, 1000+$f->getAscent()/2);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFShape->setRightFill

(no version information, might be only in CVS)

SWFShape->setRightFill -- Sets right rasterizing color

Description

void **swfshape->setrightfill** (swfgradient fill)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

void **swfshape->setrightfill** (int red, int green, int blue [, int a])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

See also **swfshape->setleftfill()**.

Shortcut for *swfshape->setrightfill($s->addfill($r, $g, $b [, $a]));*.

SWFShape

(PHP 4 >= 4.0.5)

SWFShape -- Creates a new shape object

Description

SWFShape **swfshape** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfshape() creates a new shape object.

SWFShape has the following methods : **swfshape->setline()**, **swfshape->addfill()**, **swfshape->setleftfill()**, **swfshape->setrightfill()**, **swfshape->movepento()**, **swfshape->movepen()**, **swfshape->drawlineto()**, **swfshape->drawline()**, **swfshape->drawcurveto()** and **swfshape->drawcurve()**.

This simple example will draw a big red elliptic quadrant.
Ejemplo 1. swfshape() example

```
<?php
  $s = new SWFShape();
  $s->setLine(40, 0x7f, 0, 0);
  $s->setRightFill($s->addFill(0xff, 0, 0));
  $s->movePenTo(200, 200);
  $s->drawLineTo(6200, 200);
  $s->drawLineTo(6200, 4600);
  $s->drawCurveTo(200, 4600, 200, 200);

  $m = new SWFMovie();
  $m->setDimension(6400, 4800);
  $m->setRate(12.0);
  $m->add($s);
  $m->nextFrame();

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

swfsprite->add

(no version information, might be only in CVS)

swfsprite->add -- Adds an object to a sprite

Description

void **swfsprite->add** (resource object)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfsprite->add() adds a **swfshape()**, a **swfbutton()**, a **swftext()**, a **swfaction()** or a **swfsprite()** object.

For displayable types (**swfshape()**, **swfbutton()**, **swftext()**, **swfaction()** or **swfsprite()**), this returns a handle to the object in a display list.

SWFSprite->nextframe

(no version information, might be only in CVS)

SWFSprite->nextframe -- Moves to the next frame of the animation

Description

void **swfsprite->nextframe** (void)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfsprite->setframes() moves to the next frame of the animation.

SWFSprite->remove

(no version information, might be only in CVS)

SWFSprite->remove -- Removes an object to a sprite

Description

void **swfsprite->remove** (resource object)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfsprite->remove() remove a **swfshape()**, a **swfbutton()**, a **swftext()**, a **swfaction()** or a **swfsprite()** object from the sprite.

SWFSprite->setframes

(no version information, might be only in CVS)

SWFSprite->setframes -- Sets the total number of frames in the animation

Description

void **swfsprite->setframes** (int numberoffframes)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfsprite->setframes() sets the total number of frames in the animation to *numberoffframes*.

SWFSprite

(PHP 4 >= 4.0.5)

SWFSprite -- Creates a movie clip (a sprite)

Description

SWFSprite **swfsprite** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swfsprite() are also known as a "movie clip", this allows one to create objects which are animated in their own timelines. Hence, the sprite has most of the same methods as the movie.

swfsprite() has the following methods : **swfsprite->add()**, **swfsprite->remove()**, **swfsprite->nextframe()** and **swfsprite->setframes()**.

This simple example will spin gracefully a big red square.

Ejemplo 1. swfsprite() example

```
<?php
  $s = new SWFShape();
  $s->setRightFill($s->addFill(0xff, 0, 0));
  $s->movePenTo(-500, -500);
  $s->drawLineTo(500, -500);
  $s->drawLineTo(500, 500);
  $s->drawLineTo(-500, 500);
  $s->drawLineTo(-500, -500);

  $p = new SWFSprite();
  $i = $p->add($s);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();

  $m = new SWFMovie();
  $i = $m->add($p);
  $i->moveTo(1500, 1000);
  $i->setName("blah");

  $m->setBackground(0xff, 0xff, 0xff);
  $m->setDimension(3000, 2000);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFText->addString

(no version information, might be only in CVS)

SWFText->addString -- Draws a string

Description

void **swftext->addstring** (string string)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swftext->addstring() draws the string *string* at the current pen (cursor) location. Pen is at the baseline of the text; i.e., ascending text is in the -y direction.

SWFText->getWidth

(no version information, might be only in CVS)

SWFText->getWidth -- Computes string's width

Description

void **swftext->getwidth** (string string)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swftext->addstring() returns the rendered width of the *string* string at the text object's current font, scale, and spacing settings.

SWFText->moveTo

(no version information, might be only in CVS)

SWFText->moveTo -- Moves the pen

Description

void **swftext->moveto** (int x, int y)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swftext->moveto() moves the pen (or cursor, if that makes more sense) to (*x*,*y*) in text object's coordinate space. If either is zero, though, value in that dimension stays the same. Annoying, should be fixed.

SWFText->setColor

(no version information, might be only in CVS)

SWFText->setColor -- Sets the current font color

Description

void **swftext->setcolor** (int red, int green, int blue [, int a])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swftext->setspacing() changes the current text color. Default is black. I think. Color is represented using the RGB system.

SWFText->setFont

(no version information, might be only in CVS)

SWFText->setFont -- Sets the current font

Description

void **swftext->setfont** (string font)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swftext->setfont() sets the current font to *font*.

SWFText->setHeight

(no version information, might be only in CVS)

SWFText->setHeight -- Sets the current font height

Description

void **swftext->setheight** (int height)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swftext->setheight() sets the current font height to *height*. Default is 240.

SWFText->setSpacing

(no version information, might be only in CVS)

SWFText->setSpacing -- Sets the current font spacing

Description

void **swftext->setspacing** (float spacing)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swftext->setspacing() sets the current font spacing to *spacingspacing*. Default is 1.0. 0 is all of the letters written at the same point. This doesn't really work that well because it inflates the advance across the letter, doesn't add the same amount of spacing between the letters. I should try and explain that better, prolly. Or just fix the damn thing to do constant spacing. This was really just a way to figure out how letter advances work, anyway.. So nyah.

SWFText

(PHP 4 >= 4.0.5)

SWFText -- Creates a new SWFText object

Description

SWFText **swftext** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swftext() creates a new SWFText object, fresh for manipulating.

SWFText has the following methods : **swftext->setfont()**, **swftext->setheight()**, **swftext->setspacing()**, **swftext->setcolor()**, **swftext->moveto()**, **swftext->addstring()** and **swftext->getwidth()**.

This simple example will draw a big yellow "PHP generates Flash with Ming" text, on white

background.

Ejemplo 1. swftext() example

```
<?php
  $f = new SWFFont("Techno.fdb");
  $t = new SWFText();
  $t->setFont($f);
  $t->moveTo(200, 2400);
  $t->setColor(0xff, 0xff, 0);
  $t->setHeight(1200);
  $t->addString("PHP generates Flash with Ming!!");

  $m = new SWFMovie();
  $m->setDimension(5400, 3600);

  $m->add($t);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFTextField->addstring

(no version information, might be only in CVS)

SWFTextField->addstring -- Concatenates the given string to the text field

Description

void **swftextfield->addstring** (string string)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swftextfield->setname() concatenates the string *string* to the text field.

SWFTextField->align

(no version information, might be only in CVS)

SWFTextField->align -- Sets the text field alignment

Description

void **swftextfield->align** (int alignement)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swftextfield->align() sets the text field alignment to *alignement*. Valid values for *alignement* are :

SWFTEXTFIELD_ALIGN_LEFT, SWFTEXTFIELD_ALIGN_RIGHT, SWFTEXTFIELD_ALIGN_CENTER and SWFTEXTFIELD_ALIGN_JUSTIFY.

SWFTextField->setbounds

(no version information, might be only in CVS)

SWFTextField->setbounds -- Sets the text field width and height

Description

void **swftextfield->setbounds** (int width, int height)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swftextfield->setbounds() sets the text field width to *width* and height to *height*. If you don't set the bounds yourself, Ming makes a poor guess at what the bounds are.

SWFTextField->setcolor

(no version information, might be only in CVS)

SWFTextField->setcolor -- Sets the color of the text field

Description

void **swftextfield->setcolor** (int red, int green, int blue [, int a])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swftextfield->setcolor() sets the color of the text field. Default is fully opaque black. Color is represented using RGB system.

SWFTextField->setFont

(no version information, might be only in CVS)

SWFTextField->setFont -- Sets the text field font

Description

void **swftextfield->setfont** (string font)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swftextfield->setfont() sets the text field font to the [browser-defined?] *font* font.

SWFTextField->setHeight

(no version information, might be only in CVS)

SWFTextField->setHeight -- Sets the font height of this text field font

Description

void **swftextfield->setheight** (int height)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swftextfield->setheight() sets the font height of this text field font to the given height *height*. Default is 240.

SWFTextField->setindentation

(no version information, might be only in CVS)

SWFTextField->setindentation -- Sets the indentation of the first line

Description

void **swftextfield->setindentation** (int width)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swftextfield->setindentation() sets the indentation of the first line in the text field, to *width*.

SWFTextField->setLeftMargin

(no version information, might be only in CVS)

SWFTextField->setLeftMargin -- Sets the left margin width of the text field

Description

void **swftextfield->setleftmargin** (int width)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swftextfield->setleftmargin() sets the left margin width of the text field to *width*. Default is 0.

SWFTextField->setLineSpacing

(no version information, might be only in CVS)

SWFTextField->setLineSpacing -- Sets the line spacing of the text field

Description

void **swftextfield->setlinespacing** (int height)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swftextfield->setlinespacing() sets the line spacing of the text field to the height of *height*. Default is 40.

SWFTextField->setMargins

(no version information, might be only in CVS)

SWFTextField->setMargins -- Sets the margins width of the text field

Description

void **swftextfield->setmargins** (int left, int right)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swftextfield->setmargins() set both margins at once, for the man on the go.

SWFTextField->setname

(no version information, might be only in CVS)

SWFTextField->setname -- Sets the variable name

Description

void **swftextfield->setname** (string name)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swftextfield->setname() sets the variable name of this text field to *name*, for form posting and action scripting purposes.

SWFTextField->setrightMargin

(no version information, might be only in CVS)

SWFTextField->setrightMargin -- Sets the right margin width of the text field

Description

void **swftextfield->setrightmargin** (int width)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swftextfield->setrightmargin() sets the right margin width of the text field to *width*. Default is 0.

SWFTextField

(PHP 4 >= 4.0.5)

SWFTextField -- Creates a text field object

Description

SWFTextField **swftextfield** ([int flags])

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

swftextfield() creates a new text field object. Text Fields are less flexible than **swftext()** objects- they can't be rotated, scaled non-proportionally, or skewed, but they can be used as form entries, and they can use browser-defined fonts.

The optional flags change the text field's behavior. It has the following possibles values :

- SWFTEXTFIELD_DRAWBOX draws the outline of the textfield
- SWFTEXTFIELD_HASLENGTH
- SWFTEXTFIELD_HTML allows text markup using HTML-tags
- SWFTEXTFIELD_MULTILINE allows multiple lines
- SWFTEXTFIELD_NOEDIT indicates that the field shouldn't be user-editable
- SWFTEXTFIELD_NOSELECT makes the field non-selectable
- SWFTEXTFIELD_PASSWORD obscures the data entry
- SWFTEXTFIELD_WORDWRAP allows text to wrap

Flags are combined with the bitwise OR operation. For example,

```
<?php
$t = newSWFTextField(SWFTEXTFIELD_PASSWORD | SWFTEXTFIELD_NOEDIT);
?>
```

creates a totally useless non-editable password field.

SWFTextField has the following methods : **swftextfield->setfont()**, **swftextfield->setbounds()**, **swftextfield->align()**, **swftextfield->setheight()**, **swftextfield->setleftmargin()**, **swftextfield->setrightmargin()**, **swftextfield->setmargins()**, **swftextfield->setindentation()**, **swftextfield->setlinespacing()**, **swftextfield->setcolor()**, **swftextfield->setname()** and **swftextfield->addstring()**.

LXXIII. Funciones de Miscelánea

Introducción

Estas funciones fueron colocadas aquí debido a que no parecen ajustarse a ninguna otra categoría.

Requirimientos

No se necesitan bibliotecas externas para construir esta extensión

---

Instalación

No se necesita ninguna instalación para usar estas funciones, son parte del núcleo de PHP.

Configuración en tiempo de ejecución

El comportamiento de estas funciones está afectado por los valores definidos en `php.ini`.

Tabla 1. Opciones Varias de Configuración

Nombre	Predeterminado	Modificable
ignore_user_abort	"0"	PHP_INI_ALL
highlight.string	#DD0000	PHP_INI_ALL
highlight.comment	#FF9900	PHP_INI_ALL
highlight.keyword	#007700	PHP_INI_ALL
highlight.bg	#FFFFFF	PHP_INI_ALL
highlight.default	#0000BB	PHP_INI_ALL
highlight.html	#000000	PHP_INI_ALL
browscap	NULL	PHP_INI_SYSTEM

Para más detalles sobre las constantes PHP_INI_* y su definición, vea **ini_set()**.

A continuación se presenta una corta explicación de las directivas de configuración.

ignore_user_abort **boolean**

> **TRUE** por defecto. Si se modifica a **FALSE**, los scripts terminarán tan pronto intenten generar alguna salida despuén de que el cliente haya abortado su conexión.
>
> Vea también **ignore_user_abort()**.

highlight.bg **string**, *highlight.comment* **string**, *highlight.default* **string**, *highlight.html* **string**, *highlight.keyword* **string**, *highlight.string* **string**

> Colores para el modo de Resaltado de Sintaxis. Cualquier cosa que sea aceptable en <font color="??????"> debería funcionar.

browscap **string**

> Nombre (p.ej.: `browscap.ini`) y ubicación del archivo de capacidades del navegador. Vea también **get_browser()**.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

CONNECTION_ABORTED (**integer**)

CONNECTION_NORMAL (**integer**)

CONNECTION_TIMEOUT (**integer**)

Tabla de contenidos

connection_aborted

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

connection_aborted -- Devuelve **TRUE** si el cliente está desconectado

Descripción

int **connection_aborted** (void)

Devuelve **TRUE** si el cliente está desconectado. Vea la descripción de la Gestión de la Conexión en el capítulo Características para una explicación completa.

connection_status

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

connection_status -- Devuelve el estado de la conexión en un campo de bits

Descripción

int **connection_status** (void)

Devuelve el estado de la conexión en un campo de bits. Vea la descripción de la Gestión de la Conexión en el capítulo Características para una explicación completa.

connection_timeout

(PHP 3>= 3.0.7, PHP 4 <= 4.0.4)

connection_timeout -- Devolver **TRUE** si el script ha alcanzado su tiempo de espera máximo

Descripción

bool **connection_timeout** (void)

Devuelve **TRUE** si el script ha alcanzado su tiempo de espera máximo.

Obsoleta
Esta función se considera obsoleta, y no existe siquiera a partir de 4.0.5.

Vea la descripción de Gestión de Conexión en el capítulo Características para una explicación completa.

Vea también **connection_status()**.

constant

(PHP 4 >= 4.0.4, PHP 5)

constant -- Devuelve el valor de una constante

Descripción

mixed **constant** (string nombre)

constant() devolverá el valor de la constante indicada por *nombre*.

constant() es útil si necesita recuperar el valor de una constante, pero no conoce su nombre. Esto quiere decir, si está almacenada en una variable o es devuelta por una función.

Ejemplo 1. Ejemplo de constant()

```
<?php

define("TAMMAX", 100);

echo TAMMAX;
echo constant("TAMMAX"); // igual que la linea anterior

?>
```

Vea también **define()**, **defined()** y la sección sobre Constantes.

define

(PHP 3, PHP 4 , PHP 5)

define -- Define una constante con nombre.

Descripción

int **define** (string name, mixed value [, int case_insensitive])

Define una constante con nombre, que es similar a una variable, excepto que:

- Las constantes no tienen un símbolo dólar '$' precediéndolas;
- Las constantes son accesibles desde cualquier lugar sin tener en cuenta las reglas de ámbito de las variables.
- Las constantes no pueden ser redefinidas o iniciadas una vez que han sido establecidas, y
- Las constantes sólo pueden evaluar valores escalares

El nombre de la constante se da en *name* (nombre); el valor se da en *value* (valor).

El tercer parámetro opcional *case_insensitive* también se encuentra disponible. Si se da el valor *1*, la constante se definirá no dintinguiendo mayúsculas/minusculas. El comportamiento por defecto es si distinguir; i.e. CONSTANT y Constant representan valores diferentes.

Ejemplo 1. Definición de Constantes

```
<?php
define("CONSTANT", "Hello world.");
echo CONSTANT; // outputs "Hello world."
?>
```

define() devuelve **`TRUE`** en caso de éxito y **`FALSE`** si ocurre un error.

Véase también **defined()** y la sección Constantes.

defined

(PHP 3, PHP 4 , PHP 5)

defined -- Comprueba que una constante con nombre dada existe.

Descripción

int **defined** (string name)

Devuelve **TRUE** si la constante con nombre dada en *name* (nombre) ha sido definida, **FALSE** en otro caso.

Véase también **define()** y la sección Constantes.

die

(no version information, might be only in CVS)

die -- Envía a la salida un mensaje y finaliza el script actual

Descripción

void **die** (string message)

Esta construcción del lenguaje envía a la salida un mensaje y finaliza la ejecución del script. No devuelve nada.

Ejemplo 1. Ejemplo die

```
<?php
$filename = '/path/to/data-file';
$file = fopen($filename, 'r')
  or die "unable to open file ($filename)";
?>
```

eval

(PHP 3, PHP 4, PHP 5)

eval -- Evalúa una cadena de carácteres como código PHP

Descripción

void **eval** (string code_str)

eval() evalúa la cadena de carácteres dada en *code_str* como código PHP. Entre otras cosas, ésto puede ser útil para almacenar código en un campo de texto de base de datos para una ejecución posterior.

Hay algunos aspectos a tener en cuenta cuando se utiliza **eval()**. Recuerde que la cadena de carácteres pasada debe ser código PHP válido, incluyendo aspectos como sentencias de terminación con un punto y coma para que el parser no finalice en la línea después de **eval()**, y secuencias de formato correctas en *code_str*.

Recuerde también que las variables a las que se les da valor en **eval()** retendrán estos valores posteriormente en el script principal.

Ejemplo 1. Ejemplo eval() - fusión en un único texo

```
<?php
$string = 'cup';
$name = 'coffee';
$str = 'This is a $string with my $name in it.<br>';
echo $str;
eval( "\$str = \"$str\";" );
echo $str;
?>
```

El ejemplo anterior mostrará:

```
This is a $string with my $name in it.
This is a cup with my coffee in it.
```

exit

(PHP 3, PHP 4, PHP 5)

exit -- Finaliza el script actual

Descripción

void **exit** (void)

Esta construcción del lenguaje finaliza la ejecución del script. No devuelve nada.

get_browser

(PHP 3, PHP 4 , PHP 5)

get_browser -- Indica las capacidades del navegador del usuario

Descripción

object **get_browser** ([string agente_usuario [, bool matriz_retorno]])

get_browser() intenta determinar las capacidades del navegador del usuario. Para ello consulta el archivo de información del navegador, `browscap.ini`. Por defecto, se utiliza el valor de *$_SERVER["HTTP_USER_AGENT"]*; sin embargo, puede alterar este comportamiento (es decir, consultar la información de otro navegador) pasando el parámetro opcional *agente_usuario* a **get_browser()**.

La información se devuelve en un **object**, el cual contendrá varios elementos de datos que representan, por ejemplo, los números de versión mayor y menor del navegador y la cadena ID; valores **TRUE**/**FALSE** para características como los frames, JavaScript, y cookies; y así sucesivamente. A partir de PHP 4.3.2, si el parámetro opcional *matriz_retorno* es **TRUE**, esta función devuelve un valor **array** en lugar de **object**. Puede evitar el parámetro *agente_usuario* con el valor **NULL**.

Aunque `browscap.ini` contiene información sobre muchos navegadores, depende de actualizaciones del usuario para mantener la base de datos al día. El formato del archivo es bastante auto-explicativo.

El siguiente ejemplo muestra como se puede listar toda la información disponible sobre el navegador del usuario.

Ejemplo 1. Ejemplo de get_browser()

```
<?php
echo $_SERVER['HTTP_USER_AGENT'] . "<hr />\n";

$navegador = get_browser();

foreach ($navegador as $nombre => $valor) {
    echo "<b>$nombre</b> $valor <br />\n";
}

?>
```

La salida del script anterior lucirá algo como:

```
Mozilla/4.5 [en] (X11; U; Linux 2.2.9 i586)<hr />
<b>browser_name_pattern:</b> Mozilla/4\.5.*<br />
<b>parent:</b> Netscape 4.0<br />
<b>platform:</b> Linux<br />
<b>majorver:</b> 4<br />
<b>minorver:</b> 5<br />
<b>browser:</b> Netscape<br />
<b>version:</b> 4<br />
<b>frames:</b> 1<br />
<b>tables:</b> 1<br />
<b>cookies:</b> 1<br />
<b>backgroundsounds:</b> <br />
<b>vbscript:</b> <br />
<b>javascript:</b> 1<br />
<b>javaapplets:</b> 1<br />
<b>activexcontrols:</b> <br />
<b>beta:</b> <br />
<b>crawler:</b> <br />
<b>authenticodeupdate:</b> <br />
<b>msn:</b> <br />
```

Para que ésto funcione, su opción de configuración browscap en `php.ini` debe apuntar a la ubicación correcta del archivo `browscap.ini` en su sistema. `browscap.ini` no hace parte de la distribución de PHP, pero puede encontrar un archivo browscap.ini actualizado aquí. Por defecto, la directiva browscap se encuentra comentada.

El valor *cookies* simplemente quiere decir que el navegador mismo tiene la capacidad de aceptar cookies y no quiere decir que el usuario haya habilitado el navegador para que acepte cookies o no. La única manera de probar si las cookies son aceptadas es definir una con **setcookie()**, recargar, y chequear el valor.

> **Nota:** En versiones anteriores a PHP 4.0.6, tendrá que pasar el agente de usuario a través del parámetro opcional *agente_usuario* si el valor de la directiva register_globals es *off*. En este caso, debe pasar *$HTTP_SERVER_VARS['HTTP_USER_AGENT']*.

highlight_file

(PHP 4 , PHP 5)

highlight_file -- Resaltado de sintaxis de un archivo

Descripción

mixed **highlight_file** (string nombre_archivo [, bool devolver])

La función **highlight_file()** imprime una versión con resaltado de sintaxis del código contenido en

nombre_archivo usando los colores definidos en el resaltador de sintaxis incorporado de PHP.

Si el segundo parámetro *devolver* es definido a **TRUE** entonces **highlight_file()** devolverá el código resaltado en lugar de imprimirlo. Si el segundo parámetro no es definido a **TRUE** entonces **highlight_file()** devolverá **TRUE** si tiene éxito, o **FALSE** en caso de fallo.

> **Nota:** El parámetro *devolver* apareció en PHP 4.2.0. Antes de esta versión, tenía el comportamiento predeterminado, que es **FALSE**

Atención
Debe tenerse cuidado cuando se usan las funciones **show_source()** y **highlight_file()** para asegurarse de que no se revele inadvertidamente información sensible, tal como contraseñas o cualquier otro tipo de información que pueda crear un riesgo potencial de seguridad.

> **Nota:** A partir de PHP 4.2.1 esta función es influenciada también por safe_mode y open_basedir.

Para crear una URL que pueda resaltar el código de cualquier script que le sea pasado, haremos uso de la directiva "ForceType" en Apache para generar un agradable patrón URL, y usaremos la función **highlight_file()** para mostrar un listado de código bien presentado.

En su `httpd.conf` puede agregar lo siguiente:

Ejemplo 1. Creación de una URL para resaltar código fuente

```
<Location /source>
    ForceType application/x-httpd-php
</Location>
```

Y luego cree un archivo llamado `source` y colóquelo en su directorio raíz de documentos web.

```
<html>
<head>
<title>Presentaci&oacute;n de C&oacute;digo Fuente</title>
</head>
<body bgcolor="white">
<?php
    $script = getenv('SCRIPT_FILENAME');
    if (!$script) {
        echo "<br /><b>ERROR: Se necesita el Nombre del Script</b><br />";
    } else {
        if (ereg("(\\.php|\\.inc)$",$script)) {
            echo "<h1>Fuente de: " . getenv("PATH_INFO") . "</h1>\n<hr />\n";
            highlight_file($script);
        } else {
            echo "<h1>ERROR: Solo se permiten scripts o archivos de " .
                 "inclusion PHP</h1>";
        }
    }
    echo "<hr />Procesado: " . date("Y/M/d H:i:s", time());
?>
</BODY>
</HTML>
```

Luego puede usar una URL como la siguiente para mostrar una versión en colores de un script ubicado en "/ruta/hacia/script.php" en su sitio web.

```
http://www.example.com/source/ruta/hacia/script.php
```

Vea también **highlight_string()**.

highlight_string

(PHP 4 , PHP 5)

highlight_string -- Resaltado de sintaxis de una cadena

Descripción

mixed **highlight_string** (string cadena [, bool devolver])

La función **highlight_string()** imprime una versión con resaltado de código de *cadena* usando los colores definidos en el resaltador de código incorporado de PHP.

Si el segundo parámetro *devolver* es definido a **TRUE** entonces **highlight_string()** devolverá el código resaltado como una cadena en lugar de imprimirlo. Si el segundo parámetro no es definido a **TRUE** entonces **highlight_string()** devolverá **TRUE** de tener éxito, o **FALSE** en caso de fallo.

Ejemplo 1. Ejemplo de highlight_string()

```
<?php
highlight_string('<?php phpinfo(); ?>');
?>
```

El anterior ejemplo producirá la siguiente salida (en PHP 4):

```
<code><font color="#000000">
<font color="#0000BB">&lt;?php phpinfo</font><font color="#007700">();
</font><\font color="#0000BB">?&gt;</font>
</font>
</code>
```

El anterior ejemplo producirá la siguiente salida (en PHP 5):

```
<code><span style="color: #000000">
<span style="color: #0000BB">&lt;?php phpinfo</span><span
style="color: #007700\">(); </span><span style="color:
#0000BB">?&gt;</span>
</span>
</code>
```

> **Nota:** El parámetro *devolver* apareció en PHP 4.2.0. Antes de esta versión, su comportamiento era el predeterminado, el cual es **FALSE**.

Vea también **highlight_file()**.

ignore_user_abort

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

ignore_user_abort -- Establece si la desconexión de un cliente debe suspender la ejecución del script

Descripción

int **ignore_user_abort** ([int setting])

Esta función establece si la desconexión de un cliente debe provocar la suspensión del script. Devolverá el valor previo y puede ser llamada sin argumentos para devolver el valor actual y no cambiarlo. Véase la sección sobre la Gestión de la Conexión en el capítulo Características para una

descripción completa de la gestión de la conexión en PHP.

pack

(PHP 3, PHP 4 , PHP 5)

pack -- empaqueta datos en una cadena binaria

Descripción

string **pack** (string format [, mixed args])

Empaqueta los argumentos dados en una cadena binaria siguiendo el formato *format*. Devuelve la cadena binaria que contiene los datos.

El concepto de esta función fue tomado de Perl y todos los códigos de formateo realizan la misma función. La cadena de formato consiste en códigos de formato seguidos por un argumento opcional de repetición. El argumento de repetición puede ser un valor entero o * para repetir hasta el fin de la entrada de datos. Para a, A, h, H la cuenta de repetición representa cuántos caracteres se toman de un argumento de datos, para @ es la posición absoluta donde poner los datos siguientes, para todo lo demás la cuenta de repetición especifica cuántos argumentos de datos se toman y empaquetan en la cadena binaria resultante. Actualmente están implementados:

- a cadena rellena de NUL
- A cadena rellena de ESPACIOS
- h cadena Hex, primero el medio byte inferior
- H cadena Hex, primero el medio byte superior
- c signed (con signo) char
- C unsigned (sin signo) char
- s signed short (siempre 16 bits, distribución de bytes de la máquina)
- S unsigned short (siempre 16 bits, distribución de bytes de la máquina)
- n unsigned short (siempre 16 bits, distribución de bytes gran endian)
- v unsigned short (siempre 16 bits, distribución de bytes pequeño endian)
- i signed integer (distribución de bytes y tamaños dependientes de la máquina)
- I unsigned integer (distribución de bytes y tamaños dependientes de la máquina)
- l signed long (siempre 32 bits, distribución de bytes de la máquina)
- L unsigned long (siempre 32 bits, distribución de bytes de la máquina)
- N unsigned long (siempre 32 bits, distribución de bytes gran endian)

- V unsigned long (siempre 32 bits, distribución de bytes pequeño endian)
- f float (representación y tamaño dependientes de la máquina)
- d double (representación y tamaño dependientes de la máquina)
- x byte NUL
- X Un byte hacia atrás
- @ relleno con NUL en la posición absoluta

Ejemplo 1. cadena de formato para pack

```
$binarydata = pack("nvc*", 0x1234, 0x5678, 65, 66);
```

La cadena binaria resultante tendrá 6 bytes de longitud y contendrá la secuencia de bytes 0x12, 0x34, 0x78, 0x56, 0x41, 0x42.
Adviértase que la distinción entre valores signed (con signo) y unsigned (sin signo) sólo afecta a la función **unpack()**, ya que la función **pack()** da el mismo resultado para códigos de formato con signo y sin signo.

Nótese también que internamente PHP almacena valores enteros como valores con signo de un tamaño dependiente de la máquina. Si le da un valor entero sin signo demasiado grande para ser almacenado, será convertido a un double (doble), lo que a menudo produce resultados no deseados.

php_check_syntax

(PHP 5)

php_check_syntax -- Check the PHP syntax of (and execute) the specified file

Descripción

bool **php_check_syntax** (string file_name [, string &error_message])

The **php_check_syntax()** function performs a syntax (lint) check on the specified *filename* testing for scripting errors. This is similar to using **php -l** from the commandline except **php_check_syntax()** will execute (but not output) the checked *file_name*. For example, if a function is defined in *file_name*, this defined function will be available to the file that executed **php_check_syntax()**, but output from *file_name* will be suppressed.

Lista de parámetros

file_name

The name of the file being checked.

error_message

If the *error_message* parameter is used, it will contain the error message generated by the syntax check. *error_message* is passed by reference.

Valores retornados

Returns **TRUE** if the lint check passed, and **FALSE** if the link check failed or if *file_name* cannot be opened.

Registro de cambios

Versión	Descripción
5.0.3	Calling **exit()** after **php_check_syntax()** resulted in a Segfault.
5.0.1	*error_message* is passed by reference.

Ejemplos

Ejemplo 1. php_check_syntax() example

```
<?php

$error_message = "";
$filename = "./tests.php";

if(!php_check_syntax($filename, $error_message)) {
   printf("Errors were found in the file %s:\n\n%s\n", $filename, $error_message);
} else {
   printf("The file %s contained no syntax errors.", $filename);
}

?>
```

El resultado del ejemplo seria algo similar a:

```
Errors were found in the file ./tests.php:

parse error, unexpected T_STRING in /tmp/tests.php on line 81
```

Ver también

include()
is_readable()

php_strip_whitespace

(PHP 5)

php_strip_whitespace -- Return source with stripped comments and whitespace

Descripción

string **php_strip_whitespace** (string filename)

Returns the PHP source code in *filename* with PHP comments and whitespace removed. This may be useful for determining the amount of actual code in your scripts compared with the amount of comments. This is similar to using **php -w** from the commandline.

> **Nota:** This function works as described as of PHP 5.0.1. Before this it would only return an empty string. For more information on this bug and its prior behavior, see bug report #29606.

Valores retornados

The stripped source code will be returned on success, or an empty string on failure.

Ejemplos

Ejemplo 1. php_strip_whitespace() example

```
<?php
// PHP comment here

/*
 * Another PHP comment
 */

echo        php_strip_whitespace(__FILE__);
// Newlines are considered whitespace, and are removed too:
do_nothing();
?>
```

El resultado del ejemplo seria:

```
<?php
 echo php_strip_whitespace(__FILE__); do_nothing(); ?>
```

Notice the PHP comments are gone, as are the whitespace and newline after the first echo statement.

show_source

show_source -- Alias de **highlight_file()**

Descripción

Esta función es un alias de **highlight_file()**.

sleep

(PHP 3, PHP 4 , PHP 5)

sleep -- Ejecución retardada

Descripción

void **sleep** (int seconds)

La función sleep retarda la ejecución del programa durante el número de *seconds* (segundos) dado.

Véase también **usleep()**.

time_nanosleep

(PHP 5)

time_nanosleep -- Delay for a number of seconds and nanoseconds

Descripción

mixed **time_nanosleep** (int seconds, int nanoseconds)

Delays program execution for the given number of *seconds* and *nanoseconds*.

seconds must be a positive integer, and *nanoseconds* must be a positive integer less than 1 billion.

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

If the delay was interrupted by a signal, an associative array will be returned with the components:

- *seconds* - number of seconds remaining in the delay
- *nanoseconds* - number of nanoseconds remaining in the delay

Ver también

sleep(), **usleep()**, **set_time_limit()**

Ejemplos

Ejemplo 1. time_nanosleep() example

```
<?php
// Careful! This won't work as expected if an array is returned
if (time_nanosleep(0, 500000000)) {
    echo "Slept for half a second.\n";
}

// This is better:
if (time_nanosleep(0, 500000000) === true) {
    echo "Slept for half a second.\n";
}

// And this is the best:
$nano = time_nanosleep(2, 100000);

if ($nano === true) {
    echo "Slept for 2 seconds, 100 milliseconds.\n";
} elseif ($nano === false) {
    echo "Sleeping failed.\n";
} elseif (is_array($nano)) {
    $seconds = $nano['seconds'];
    $nanoseconds = $nano['nanoseconds'];
    echo "Interrupted by a signal.\n";
    echo "Time remaining: $seconds seconds, $nanoseconds nanoseconds.";
}
?>
```

Nota: Esta función no está implementada en plataformas Windows.

uniqid

(PHP 3, PHP 4 , PHP 5)

uniqid -- Genera un id único.

Descripción

int **uniqid** (string prefix [, boolean lcg])

uniqid() devuelve un identificador único con un prefijo basado en la hora actual en microsegundos. El prefijo puede ser práctico por ejemplo si se generan identificadores simultáneamente en varios host que pueden haber generado el identificador en el mismo microsegundo. *prefix* (prefijo) puede ser de hasta 114 carácteres de longitud.

Si el parámetro opcional *lcg* es **TRUE**, **uniqid()** añadirá entropía "LCG combinada" al final del valor devuelto, que hará el resultado más único.

Con un *prefix* (prefijo) vacío, la cadena devuelta tendrá una longitud de 13 carácteres. Si *lcg* es **TRUE**, tendrá 23 carácteres.

Nota: El parámetro *lcg* está disponible sólo en PHP 4 y PHP 3.0.13 y posteriores.

Si necesita un identificador único o testigo, y tiene la intención de hacer público ese testigo al usuario por medio de una red (i.e. cookies de sesión) se recomienda que utilice algo parecido a estas líneas

```
$token = md5(uniqid("")); // no random portion
$better_token = md5(uniqid(rand())); // better, difficult to guess
```

Esto creará un identificador de 32 carácteres (un número hexadecimal de 128 bits) que es extremadamente difícil de predecir.

unpack

(PHP 3, PHP 4 , PHP 5)

unpack -- desempaqueta datos de una cadena binaria

Descripción

array **unpack** (string format, string data)

Desempaqueta datos de una cadena binaria en un array, de acuerdo al formato *format*. Devuelve un array que contiene los elementos de la cadena binaria desempaquetados.

Unpack funciona de manera ligeramente diferente a Perl, ya que los datos desempaquetados se almacenan en un array asociativo. Para conseguir ésto debe nombrar los diferentes códigos de formato y separarlos por una barra inclinada /.
Ejemplo 1. cadena de formato unpack

```
$array = unpack("c2chars/nint", $binarydata);
```
El array resultante contendrá las entradas "chars1", "chars2" y "int".
Para una explicación de los códigos de formato véase también: **pack()**

Advierta que PHP almacena internamente los valores enteros con signo. Si desempaqueta un unsigned long (largo sin signo) demasiado grande y es del mismo tamaño tal como PHP almacena internamente los valores, el resultado será un número negativo a pesar de que se especificara desempaquetamiento sin signo.

usleep

(PHP 3, PHP 4 , PHP 5)

usleep -- Retrasa la ejecución, en microsegundos

Descripción

void **usleep** (int micro_seconds)

La función usleep retrasa la ejecución del programa durante un número de *micro_seconds* (microsegundos) dado.

Véase también **sleep()**.

LXXIV. mnoGoSearch Functions

Introducción

These functions allow you to access the mnoGoSearch (former UdmSearch) free search engine. mnoGoSearch is a full-featured search engine software for intranet and internet servers, distributed under the GNU license. mnoGoSearch has a number of unique features, which makes it appropriate for a wide range of applications from search within your site to a specialized search system such as cooking recipes or newspaper search, FTP archive search, news articles search, etc. It offers full-text indexing and searching for HTML, PDF, and text documents. mnoGoSearch consists of two parts. The first is an indexing mechanism (indexer). The purpose of the indexer is to walk through HTTP, FTP, NEWS servers or local files, recursively grabbing all the documents and storing meta-data about that documents in a SQL database in a smart and effective manner. After every document is referenced by its corresponding URL, meta-data is collected by the indexer for later use in a search process. The search is performed via Web interface. C, CGI, PHP and Perl search front ends are included.

More information about mnoGoSearch can be found at http://www.mnogosearch.org/.

> **Nota:** Esta extensión no está disponible en plataformas Windows

---

Requirimientos

Download mnoGosearch from http://www.mnogosearch.org/ and install it on your system. You need at least version 3.1.10 of mnoGoSearch installed to use these functions.

Instalación

In order to have these functions available, you must compile PHP with mnoGosearch support by using the *--with-mnogosearch*option. If you use this option without specifying the path to mnoGosearch, PHP will look for mnoGosearch under `/usr/local/mnogosearch` path by default. If you installed mnoGosearch at a different location you should specify it: *--with-mnogosearch=DIR*.

> **Nota:** PHP contains built-in MySQL access library, which can be used to access MySQL. It is known that mnoGoSearch is not compatible with this built-in library and can work only with generic MySQL libraries. Thus, if you use mnoGoSearch with MySQL, during PHP configuration you have to indicate the directory of your MySQL installation, that was used during mnoGoSearch configuration, i.e. for example: *--with-mnogosearch --with-mysql=/usr*.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

`UDM_FIELD_URLID` (**integer**)

`UDM_FIELD_URL` (**integer**)

`UDM_FIELD_CONTENT` (**integer**)

`UDM_FIELD_TITLE` (**integer**)

`UDM_FIELD_KEYWORDS` (**integer**)

`UDM_FIELD_DESC` (**integer**)

`UDM_FIELD_DESCRIPTION` (**integer**)

`UDM_FIELD_TEXT` (**integer**)

`UDM_FIELD_SIZE` (**integer**)

UDM_FIELD_RATING (integer)

UDM_FIELD_SCORE (integer)

UDM_FIELD_MODIFIED (integer)

UDM_FIELD_ORDER (integer)

UDM_FIELD_CRC (integer)

UDM_FIELD_CATEGORY (integer)

UDM_FIELD_LANG (integer)

UDM_FIELD_CHARSET (integer)

UDM_PARAM_PAGE_SIZE (integer)

UDM_PARAM_PAGE_NUM (integer)

UDM_PARAM_SEARCH_MODE (integer)

UDM_PARAM_CACHE_MODE (integer)

UDM_PARAM_TRACK_MODE (integer)

UDM_PARAM_PHRASE_MODE (integer)

UDM_PARAM_CHARSET (integer)

UDM_PARAM_LOCAL_CHARSET (integer)

UDM_PARAM_BROWSER_CHARSET (integer)

UDM_PARAM_STOPTABLE (integer)

UDM_PARAM_STOP_TABLE (integer)

UDM_PARAM_STOPFILE (integer)

UDM_PARAM_STOP_FILE (integer)

UDM_PARAM_WEIGHT_FACTOR (integer)

UDM_PARAM_WORD_MATCH (integer)

UDM_PARAM_MAX_WORD_LEN (integer)

UDM_PARAM_MAX_WORDLEN (integer)

UDM_PARAM_MIN_WORD_LEN (integer)

UDM_PARAM_MIN_WORDLEN (integer)

UDM_PARAM_ISPELL_PREFIXES (integer)

UDM_PARAM_ISPELL_PREFIX (integer)

UDM_PARAM_PREFIXES (integer)

UDM_PARAM_PREFIX (integer)

UDM_PARAM_CROSS_WORDS (integer)

UDM_PARAM_CROSSWORDS (integer)

UDM_PARAM_VARDIR (integer)

UDM_PARAM_DATADIR (integer)

UDM_PARAM_HLBEG (integer)

UDM_PARAM_HLEND (integer)

UDM_PARAM_SYNONYM (integer)

UDM_PARAM_SEARCHD (integer)

UDM_PARAM_QSTRING (integer)

UDM_PARAM_REMOTE_ADDR (integer)

UDM_LIMIT_CAT (integer)

UDM_LIMIT_URL (integer)

UDM_LIMIT_TAG (integer)

UDM_LIMIT_LANG (integer)

UDM_LIMIT_DATE (integer)

UDM_PARAM_FOUND (integer)

UDM_PARAM_NUM_ROWS (integer)

UDM_PARAM_WORDINFO (integer)

UDM_PARAM_WORD_INFO (integer)

UDM_PARAM_SEARCHTIME (integer)

UDM_PARAM_SEARCH_TIME (integer)

UDM_PARAM_FIRST_DOC (integer)

UDM_PARAM_LAST_DOC (integer)

UDM_MODE_ALL (integer)

UDM_MODE_ANY (integer)

UDM_MODE_BOOL (integer)

UDM_MODE_PHRASE (integer)

UDM_CACHE_ENABLED (integer)

UDM_CACHE_DISABLED (integer)

UDM_TRACK_ENABLED (integer)

UDM_TRACK_DISABLED (integer)

UDM_PHRASE_ENABLED (integer)

UDM_PHRASE_DISABLED (integer)

UDM_CROSS_WORDS_ENABLED (integer)

UDM_CROSSWORDS_ENABLED (integer)

UDM_CROSS_WORDS_DISABLED (integer)

UDM_CROSSWORDS_DISABLED (integer)

UDM_PREFIXES_ENABLED (integer)

UDM_PREFIX_ENABLED (integer)

UDM_ISPELL_PREFIXES_ENABLED (integer)

UDM_ISPELL_PREFIX_ENABLED (integer)

UDM_PREFIXES_DISABLED (integer)

UDM_PREFIX_DISABLED (integer)

UDM_ISPELL_PREFIXES_DISABLED (integer)

UDM_ISPELL_PREFIX_DISABLED (integer)

UDM_ISPELL_TYPE_AFFIX (integer)

UDM_ISPELL_TYPE_SPELL (integer)

UDM_ISPELL_TYPE_DB (integer)

UDM_ISPELL_TYPE_SERVER (integer)

UDM_MATCH_WORD (integer)

UDM_MATCH_BEGIN (integer)

UDM_MATCH_SUBSTR (integer)

UDM_MATCH_END (integer)

Tabla de contenidos

udm_add_search_limit

(PHP 4 >= 4.0.5, PHP 5)

udm_add_search_limit -- Add various search limits

Description

bool **udm_add_search_limit** (resource agent, int var, string val)

udm_add_search_limit() adds search restrictions. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

agent - a link to Agent, received after call to **udm_alloc_agent()**.

var - defines parameter, indicating limit.

val - defines the value of the current parameter.

Possible *var* values:

- UDM_LIMIT_URL - defines document URL limitations to limit the search through subsection of the database. It supports SQL % and _ LIKE wildcards, where % matches any number of characters, even zero characters, and _ matches exactly one character. E.g. http://www.example.___/catalog may stand for http://www.example.com/catalog and http://www.example.net/catalog.

- UDM_LIMIT_TAG - defines site TAG limitations. In indexer-conf you can assign specific TAGs to various sites and parts of a site. Tags in mnoGoSearch 3.1.x are lines, that may contain metasymbols % and _. Metasymbols allow searching among groups of tags. E.g. there are links with tags ABCD and ABCE, and search restriction is by ABC_ - the search will be made among both of the tags.

- UDM_LIMIT_LANG - defines document language limitations.

- UDM_LIMIT_CAT - defines document category limitations. Categories are similar to tag feature, but nested. So you can have one category inside another and so on. You have to use two characters for each level. Use a hex number going from 0-F or a 36 base number going from 0-Z. Therefore a top-level category like 'Auto' would be 01. If it has a subcategory like 'Ford', then it would be 01 (the parent category) and then 'Ford' which we will give 01. Put those together and you get 0101. If 'Auto' had another subcategory named 'VW', then it's id would be 01 because it belongs to the 'Ford' category and then 02 because it's the next category. So it's id would be 0102. If VW had a sub category called 'Engine' then it's id would start at 01 again and it would get the 'VW' id 02 and 'Auto' id of 01, making it 010201. If you want to search for sites under that category then you pass it cat=010201 in the URL.

- UDM_LIMIT_DATE - defines limitation by date the document was modified.

  Format of parameter value: a string with first character < or >, then with no space - date in unixtime format, for example:

  Ejemplo 1.
  ```
  <?php
        Udm_Add_Search_Limit($udm, UDM_LIMIT_DATE, "&lt;908012006");
  ?>
  ```
  If > character is used, then the search will be restricted to those documents having a modification date greater than entered, if <, then smaller.

udm_alloc_agent_array

(PHP 4 >= 4.3.3, PHP 5)

udm_alloc_agent_array -- Allocate mnoGoSearch session

Description

resource **udm_alloc_agent_array** (array databases)

udm_alloc_agent_array() will create an agent with multiple database connections. The array *databases* must contain one database URL per element, analog to the first parameter of **udm_alloc_agent()**.

See also: **udm_alloc_agent()**.

udm_alloc_agent

(PHP 4 >= 4.0.5, PHP 5)

udm_alloc_agent -- Allocate mnoGoSearch session

Description

resource **udm_alloc_agent** (string dbaddr [, string dbmode])

Returns a mnogosearch agent identifier on success, **FALSE** on failure. This function creates a session with database parameters.

dbaddr - URL-style database description, with options (type, host, database name, port, user and password) to connect to SQL database. Do not matter for built-in text files support. Format for *dbaddr*: *DBType:[//[DBUser[:DBPass]@]DBHost[:DBPort]]/DBName/*. Currently supported DBType values are: mysql, pgsql, msql, solid, mssql, oracle, and ibase. Actually, it does not matter for native libraries support, but ODBC users should specify one of the supported values. If your database type is not supported, you may use *unknown* instead.

dbmode - You may select the SQL database mode of words storage. Possible values of *dbmode* are: *single*, *multi*, *crc*, or *crc-multi*. When *single* is specified, all words are stored in the same table. If *multi* is selected, words will be located in different tables depending of their lengths. "multi" mode is usually faster, but requires more tables in the database. If "crc" mode is selected, mnoGoSearch will store 32 bit integer word IDs calculated by CRC32 algorithm instead of words. This mode requires less disk space and it is faster comparing with "single" and "multi" modes. *crc-multi* uses the same storage structure with the "crc" mode, but also stores words in different tables depending on words lengths like in "multi" mode.

> **Nota:** *dbaddr* and *dbmode* must match those used during indexing.

> **Nota:** In fact this function does not open a connection to the database and thus does not check the entered login and password. Establishing a connection to the database and login/password verification is done by **udm_find()**.

udm_api_version

(PHP 4 >= 4.0.5, PHP 5)

udm_api_version -- Get mnoGoSearch API version

Description

int **udm_api_version** (void)

udm_api_version() returns the mnoGoSearch API version number. E.g. if mnoGoSearch 3.1.10 API is used, this function will return *30110*.

This function allows the user to identify which API functions are available, e.g. **udm_get_doc_count()** function is only available in mnoGoSearch 3.1.11 or later.

Ejemplo 1. udm_api_version() example

```
<?php
if (udm_api_version() >= 30111) {
    echo  "Total number of URLs in database: " . udm_get_doc_count($udm) . "<br />\n";
}
?>
```

udm_cat_list

(PHP 4 >= 4.0.6, PHP 5)

udm_cat_list -- Get all the categories on the same level with the current one

Description

array **udm_cat_list** (resource agent, string category)

Returns an array listing all categories of the same level as the current *category* in the categories tree. *agent* is the agent identifier returned by a previous call to **>udm_alloc_agent()**.

The function can be useful for developing categories tree browser.

The returned array consists of pairs. Elements with even index numbers contain the category paths, odd elements contain the corresponding category names.

```
$array[0] will contain '020300'
  $array[1] will contain 'Audi'
  $array[2] will contain '020301'
  $array[3] will contain 'BMW'
  $array[4] will contain '020302'
  $array[5] will contain 'Opel'
  ...
 etc.
```

Following is an example of displaying links of the current level in format:

```
Audi
  BMW
  Opel
  ...
```

Ejemplo 1. udm_cat_list()example

```
<?php
 $cat_list_arr = udm_cat_list($udm_agent, $cat);
 $cat_list = '';
 for ($i=0; $i<count($cat_list_arr); $i+=2) {
    $path = $cat_list_arr[$i];
    $name = $cat_list_arr[$i+1];
    $cat_list .= "<a href=\"$_SERVER[PHP_SELF]?cat=$path\">$name</a><br />";
 }
?>
```

See also **udm_cat_path()**.

udm_cat_path

(PHP 4 >= 4.0.6, PHP 5)

udm_cat_path -- Get the path to the current category

Description

array **udm_cat_path** (resource agent, string category)

Returns an array describing the path in the categories tree from the tree root to the current one, specified by *category*. *agent* is the agent identifier returned by a previous call to **>udm_alloc_agent ()**.

The returned array consists of pairs. Elements with even index numbers contain the category paths, odd elements contain the corresponding category names.

For example, the call *$array=udm_cat_path($agent, '02031D');* may return the following array:

```
$array[0] will contain ''
 $array[1] will contain 'Root'
 $array[2] will contain '02'
 $array[3] will contain 'Sport'
 $array[4] will contain '0203'
 $array[5] will contain 'Auto'
 $array[4] will contain '02031D'
 $array[5] will contain 'Ferrari'
```

Ejemplo 1. Specifying path to the current category in the following format: '> Root > Sport > Auto > Ferrari'

```
<?php
  $cat_path_arr = udm_cat_path($udm_agent, $cat);
  $cat_path = '';
  for ($i=0; $i<count($cat_path_arr); $i+=2) {
    $path = $cat_path_arr[$i];
    $name = $cat_path_arr[$i+1];
    $cat_path .= " > <a href=\"$_SERVER[PHP_SELF]?cat=$path\">$name</a> ";
  }
?>
```

See also **udm_cat_list()**.

udm_check_charset

(PHP 4 >= 4.2.0, PHP 5)

udm_check_charset -- Check if the given charset is known to mnogosearch

Description

bool **udm_check_charset** (resource agent, string charset)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

udm_check_stored

(PHP 4 >= 4.2.0, PHP 5)

udm_check_stored -- Check connection to stored

Description

int **udm_check_stored** (resource agent, int link, string doc_id)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

udm_clear_search_limits

(PHP 4 >= 4.0.5, PHP 5)

udm_clear_search_limits -- Clear all mnoGoSearch search restrictions

Description

bool **udm_clear_search_limits** (resource agent)

udm_clear_search_limits() resets defined search limitations and returns **`TRUE`**.

See also **udm_add_search_limit()**.

udm_close_stored

(PHP 4 >= 4.2.0, PHP 5)

udm_close_stored -- Close connection to stored

Description

int **udm_close_stored** (resource agent, int link)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

udm_crc32

(PHP 4 >= 4.2.0, PHP 5)

udm_crc32 -- Return CRC32 checksum of given string

Description

int **udm_crc32** (resource agent, string str)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

udm_errno

(PHP 4 >= 4.0.5, PHP 5)

udm_errno -- Get mnoGoSearch error number

Description

int **udm_errno** (resource agent)

udm_errno() returns mnoGoSearch error number, zero if no error.

agent - link to agent identifier, received after call to **udm_alloc_agent()**.

Receiving numeric agent error code.

udm_error

(PHP 4 >= 4.0.5, PHP 5)

udm_error -- Get mnoGoSearch error message

Description

string **udm_error** (resource agent)

udm_error() returns mnoGoSearch error message, empty string if no error.

agent - link to agent identifier, received after call to **udm_alloc_agent()**.

Receiving agent error message.

udm_find

(PHP 4 >= 4.0.5, PHP 5)

udm_find -- Perform search

Description

resource **udm_find** (resource agent, string query)

Returns a result link identifier on success, or **FALSE** on failure.

The search itself. The first argument - session, the next one - query itself. To find something just type words you want to find and press SUBMIT button. For example, "mysql odbc". You should not use quotes " in query, they are written here only to divide a query from other text. mnoGoSearch will find all documents that contain word "mysql" and/or word "odbc". Best documents having bigger weights will be displayed first. If you use search mode ALL, search will return documents that contain both (or more) words you entered. In case you use mode ANY, the search will return list of documents that contain any of the words you entered. If you want more advanced results you may use query language. You should select "bool" match mode in the search from.

mnoGoSearch understands the following boolean operators:

& - logical AND. For example, "mysql & odbc". mnoGoSearch will find any URLs that contain both "mysql" and "odbc".

| - logical OR. For example "mysql|odbc". mnoGoSearch will find any URLs, that contain word "mysql" or word "odbc".

~ - logical NOT. For example "mysql & ~odbc". mnoGoSearch will find URLs that contain word "mysql" and do not contain word "odbc" at the same time. Note that ~ just excludes given word from results. Query "~odbc" will find nothing!

() - group command to compose more complex queries. For example "(mysql | msql) & ~postgres". Query language is simple and powerful at the same time. Just consider query as usual boolean expression.

udm_free_agent

(PHP 4 >= 4.0.5, PHP 5)

udm_free_agent -- Free mnoGoSearch session

Description

int **udm_free_agent** (resource agent)

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

agent - link to agent identifier, received ` after call to **udm_alloc_agent()**.

Freeing up memory allocated for agent session.

udm_free_ispell_data

(PHP 4 >= 4.0.5, PHP 5)

udm_free_ispell_data -- Free memory allocated for ispell data

Description

bool **udm_free_ispell_data** (int agent)

udm_free_ispell_data() always returns **TRUE**.

agent - agent link identifier, received after call to **udm_alloc_agent()**.

> **Nota:** This function is supported beginning from version 3.1.12 of mnoGoSearch and it does not do anything in previous versions.

udm_free_res

(PHP 4 >= 4.0.5, PHP 5)

udm_free_res -- Free mnoGoSearch result

Description

bool **udm_free_res** (resource res)

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

res - a link to result identifier, received after call to **udm_find()**.

Freeing up memory allocated for results.

udm_get_doc_count

(PHP 4 >= 4.0.5, PHP 5)

udm_get_doc_count -- Get total number of documents in database

Description

int **udm_get_doc_count** (resource agent)

udm_get_doc_count() returns the number of documents in the database.

agent - link to agent identifier, received after call to **udm_alloc_agent()**.

> **Nota:** This function is supported only in mnoGoSearch 3.1.11 or later.

udm_get_res_field

(PHP 4 >= 4.0.5, PHP 5)

udm_get_res_field -- Fetch mnoGoSearch result field

Description

string **udm_get_res_field** (resource res, int row, int field)

udm_get_res_field() returns result field value on success, **FALSE** on error.

res - a link to result identifier, received after call to **udm_find()**.

row - the number of the link on the current page. May have values from 0 to *UDM_PARAM_NUM_ROWS-1*.

field - field identifier, may have the following values:

- UDM_FIELD_URL - document URL field
- UDM_FIELD_CONTENT - document Content-type field (for example, text/html).
- UDM_FIELD_CATEGORY - document category field. Use **udm_cat_path()** to get full path to current category from the categories tree root. (This parameter is available only in PHP 4.0.6 or later).
- UDM_FIELD_TITLE - document title field.
- UDM_FIELD_KEYWORDS - document keywords field (from META KEYWORDS tag).
- UDM_FIELD_DESC - document description field (from META DESCRIPTION tag).
- UDM_FIELD_TEXT - document body text (the first couple of lines to give an idea of what the document is about).
- UDM_FIELD_SIZE - document size.
- UDM_FIELD_URLID - unique URL ID of the link.
- UDM_FIELD_RATING - page rating (as calculated by mnoGoSearch).
- UDM_FIELD_MODIFIED - last-modified field in unixtime format.
- UDM_FIELD_ORDER - the number of the current document in set of found documents.
- UDM_FIELD_CRC - document CRC.

udm_get_res_param

(PHP 4 >= 4.0.5, PHP 5)

udm_get_res_param -- Get mnoGoSearch result parameters

Description

string **udm_get_res_param** (resource res, int param)

udm_get_res_param() returns result parameter value on success, **FALSE** on error.

res - a link to result identifier, received after call to **udm_find()**.

param - parameter identifier, may have the following values:

- UDM_PARAM_NUM_ROWS - number of received found links on the current page. It is equal to UDM_PARAM_PAGE_SIZE for all search pages, on the last page - the rest of links.
- UDM_PARAM_FOUND - total number of results matching the query.
- UDM_PARAM_WORDINFO - information on the words found. E.g. search for "a good book" will return "a: stopword, good:5637, book: 120"
- UDM_PARAM_SEARCHTIME - search time in seconds.
- UDM_PARAM_FIRST_DOC - the number of the first document displayed on current page.
- UDM_PARAM_LAST_DOC - the number of the last document displayed on current page.

udm_hash32

(PHP 4 >= 4.3.3, PHP 5)

udm_hash32 -- Return Hash32 checksum of gived string

Description

int **udm_hash32** (resource agent, string str)

udm_hash32() will take a string *str* and return a quite unique 32-bit hash number from it. Requires an allocated *agent*.

See also: **udm_alloc_agent()**.

udm_load_ispell_data

(PHP 4 >= 4.0.5, PHP 5)

udm_load_ispell_data -- Load ispell data

Description

bool **udm_load_ispell_data** (resource agent, int var, string val1, string val2, int flag)

udm_load_ispell_data() loads ispell data. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

agent - agent link identifier, received after call to **udm_alloc_agent()**.

var - parameter, indicating the source for ispell data. May have the following values:

After using this function to free memory allocated for ispell data, please use **udm_free_ispell_data()**, even if you use UDM_ISPELL_TYPE_SERVER mode.

The fastest mode is UDM_ISPELL_TYPE_SERVER. UDM_ISPELL_TYPE_TEXT is slower and UDM_ISPELL_TYPE_DB is the slowest. The above pattern is **TRUE** for mnoGoSearch 3.1.10 - 3.1.11. It is planned to speed up DB mode in future versions and it is going to be faster than TEXT mode.

- UDM_ISPELL_TYPE_DB - indicates that ispell data should be loaded from SQL. In this case, parameters *val1* and *val2* are ignored and should be left blank. *flag* should be equal to *1*.

  Nota: *flag* indicates that after loading ispell data from defined source it should be sorted (it is necessary for correct functioning of ispell). In case of loading ispell data from files there may be several calls to **udm_load_ispell_data()**, and there is no sense to sort data after every call, but only after the last one. Since in db mode all the data is loaded by one call, this parameter should have the value *1*. In this mode in case of error, e.g. if ispell tables are absent, the function will return **FALSE** and code and error message will be accessible through **udm_error()** and **udm_errno()**.

  Ejemplo 1. udm_load_ispell_data()example

```
<?php
if (! udm_load_ispell_data($udm, UDM_ISPELL_TYPE_DB, '', '', 1)) {
  printf("Error #%d: '%s'\n", udm_errno($udm), udm_error($udm));
  exit;
}
?>
```

- UDM_ISPELL_TYPE_AFFIX - indicates that ispell data should be loaded from file and initiates loading affixes file. In this case *val1* defines double letter language code for which affixes are loaded, and *val2* - file path. Please note, that if a relative path entered, the module looks for the file not in UDM_CONF_DIR, but in relation to current path, i.e. to the path where the script is executed. In case of error in this mode, e.g. if file is absent, the function will return **FALSE**, and an error message will be displayed. Error message text cannot be accessed through **udm_error()** and **udm_errno()**, since those functions can only return messages associated with SQL. Please, see *flag* parameter description in UDM_ISPELL_TYPE_DB.

  Ejemplo 2. udm_load_ispell_data() example

```
<?php
if ((! udm_load_ispell_data($udm, UDM_ISPELL_TYPE_AFFIX, 'en', '/opt/ispell/en.af
    (! udm_load_ispell_data($udm, UDM_ISPELL_TYPE_AFFIX, 'ru', '/opt/ispell/ru.af
    (! udm_load_ispell_data($udm, UDM_ISPELL_TYPE_SPELL, 'en', '/opt/ispell/en.di
    (! udm_load_ispell_data($udm, UDM_ISPELL_TYPE_SPELL, 'ru', '/opt/ispell/ru.di
    exit;
}
?>
```

  Nota: *flag* is equal to *1* only in the last call.

- UDM_ISPELL_TYPE_SPELL - indicates that ispell data should be loaded from file and initiates loading of ispell dictionary file. In this case *val1* defines double letter language code for which affixes are loaded, and *val2* - file path. Please note, that if a relative path entered, the module looks for the file not in UDM_CONF_DIR, but in relation to current path, i.e. to the path where the script is executed. In case of error in this mode, e.g. if file is absent, the function will return **FALSE**, and an error message will be displayed. Error message text cannot be accessed through **udm_error()** and **udm_errno()**, since those functions can only return messages associated with SQL. Please, see *flag* parameter description in UDM_ISPELL_TYPE_DB.

```
<?php
     if ((! Udm_Load_Ispell_Data($udm, UDM_ISPELL_TYPE_AFFIX, 'en', '/opt/ispell/
        (! Udm_Load_Ispell_Data($udm, UDM_ISPELL_TYPE_AFFIX, 'ru', '/opt/ispell/r
        (! Udm_Load_Ispell_Data($udm, UDM_ISPELL_TYPE_SPELL, 'en', '/opt/ispell/e
        (! Udm_Load_Ispell_Data($udm, UDM_ISPELL_TYPE_SPELL, 'ru', '/opt/ispell/r
      exit;
      }
?>
```

Nota: *flag* is equal to *1* only in the last call.

- UDM_ISPELL_TYPE_SERVER - enables spell server support. *val1* parameter indicates address of the host running spell server. *val2* ` is not used yet, but in future releases it is going to indicate number of port used by spell server. *flag* parameter in this case is not needed since ispell data is stored on spellserver already sorted.

  Spelld server reads spell-data from a separate configuration file (/usr/local/mnogosearch/etc/spelld.conf by default), sorts it and stores in memory. With clients server communicates in two ways: to indexer all the data is transferred (so that indexer starts faster), from search.cgi server receives word to normalize and then passes over to client (search.cgi) list of normalized word forms. This allows fastest, compared to db and text modes processing of search queries (by omitting loading and sorting all the spell data).

  udm_load_ispell_data() function in UDM_ISPELL_TYPE_SERVER mode does not actually load ispell data, but only defines server address. In fact, server is automatically used by **udm_find()** function when performing search. In case of errors, e.g. if spellserver is not running or invalid host indicated, there are no messages returned and ispell conversion does not work.

  Nota: This function is available in mnoGoSearch 3.1.12 or later.

  Example:

```
<?php
if (!udm_load_ispell_data($udm, UDM_ISPELL_TYPE_SERVER, '', '', 1)) {
    echo "Error loading ispell data from server<br />\n";
    exit;
}
?>
```

udm_open_stored

(PHP 4 >= 4.2.0, PHP 5)

udm_open_stored -- Open connection to stored

Description

int **udm_open_stored** (resource agent, string storedaddr)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

udm_set_agent_param

(PHP 4 >= 4.0.5, PHP 5)

udm_set_agent_param -- Set mnoGoSearch agent session parameters

Description

bool **udm_set_agent_param** (resource agent, int var, string val)

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo. Defines mnoGoSearch session parameters.

The following parameters and their values are available:

- UDM_PARAM_PAGE_NUM - used to choose search results page number (results are returned by pages beginning from 0, with UDM_PARAM_PAGE_SIZE results per page).
- UDM_PARAM_PAGE_SIZE - number of search results displayed on one page.
- UDM_PARAM_SEARCH_MODE - search mode. The following values available: UDM_MODE_ALL - search for all words; UDM_MODE_ANY - search for any word; UDM_MODE_PHRASE - phrase search; UDM_MODE_BOOL - boolean search. See **udm_find()** for details on boolean search.
- UDM_PARAM_CACHE_MODE - turns on or off search result cache mode. When enabled, the search engine will store search results to disk. In case a similar search is performed later, the engine will take results from the cache for faster performance. Available values: UDM_CACHE_ENABLED, UDM_CACHE_DISABLED.
- UDM_PARAM_TRACK_MODE - turns on or off trackquery mode. Since version 3.1.2 mnoGoSearch has a query tracking support. Note that tracking is implemented in SQL version only and not available in built-in database. To use tracking, you have to create tables for tracking support. For MySQL, use create/mysql/track.txt. When doing a search, front-end uses those tables to store query words, a number of found documents and current Unix timestamp in seconds. Available values: UDM_TRACK_ENABLED, UDM_TRACK_DISABLED.
- UDM_PARAM_PHRASE_MODE - defines whether index database using phrases ("phrase" parameter in indexer.conf). Possible values: UDM_PHRASE_ENABLED and UDM_PHRASE_DISABLED. Please note, that if phrase search is enabled (UDM_PHRASE_ENABLED), it is still possible to do search in any mode (ANY, ALL, BOOL or PHRASE). In 3.1.10 version of mnoGoSearch phrase search is supported only in

sql and built-in database modes, while beginning with 3.1.11 phrases are supported in cachemode as well.

Examples of phrase search:

"Arizona desert" - This query returns all indexed documents that contain "Arizona desert" as a phrase. Notice that you need to put double quotes around the phrase

- UDM_PARAM_CHARSET - defines local charset. Available values: set of charsets supported by mnoGoSearch, e.g. koi8-r, cp1251, ...
- UDM_PARAM_STOPFILE - Defines name and path to stopwords file. (There is a small difference with mnoGoSearch - while in mnoGoSearch if relative path or no path entered, it looks for this file in relation to UDM_CONF_DIR, the module looks for the file in relation to current path, i.e. to the path where the PHP script is executed.)
- UDM_PARAM_STOPTABLE - Load stop words from the given SQL table. You may use several StopwordTable commands. This command has no effect when compiled without SQL database support.
- UDM_PARAM_WEIGHT_FACTOR - represents weight factors for specific document parts. Currently body, title, keywords, description, url are supported. To activate this feature please use degrees of 2 in *Weight commands of the indexer.conf. Let's imagine that we have these weights:

```
URLWeight     1
BodyWeight    2
TitleWeight   4
KeywordWeight 8
DescWeight    16
```

As far as indexer uses bit OR operation for word weights when some word presents several time in the same document, it is possible at search time to detect word appearance in different document parts. Word which appears only in the body will have 00000010 aggregate weight (in binary notation). Word used in all document parts will have 00011111 aggregate weight.

This parameter's value is a string of hex digits ABCDE. Each digit is a factor for corresponding bit in word weight. For the given above weights configuration:

```
E is a factor for weight 1  (URL Weight bit)
D is a factor for weight 2  (BodyWeight bit)
C is a factor for weight 4  (TitleWeight bit)
B is a factor for weight 8  (KeywordWeight bit)
A is a factor for weight 16 (DescWeight bit)
```

Examples:

UDM_PARAM_WEIGHT_FACTOR=00001 will search through URLs only.

UDM_PARAM_WEIGHT_FACTOR=00100 will search through Titles only.

UDM_PARAM_WEIGHT_FACTOR=11100 will search through Title,Keywords,Description but not through URL and Body.

UDM_PARAM_WEIGHT_FACTOR=F9421 will search through:

Description with factor 15 (F hex)
Keywords with factor 9
Title with factor 4
Body with factor 2
URL with factor 1

If UDM_PARAM_WEIGHT_FACTOR variable is omitted, original weight value is taken to sort results. For a given above weight configuration it means that document description has a most big weight 16.

- UDM_PARAM_WORD_MATCH - word match. You may use this parameter to choose word match type. This feature works only in "single" and "multi" modes using SQL based and built-in database. It does not work in cachemode and other modes since they use word CRC and do not support substring search. Available values:

  UDM_MATCH_BEGIN - word beginning match;

  UDM_MATCH_END - word ending match;

  UDM_MATCH_WORD - whole word match;

  UDM_MATCH_SUBSTR - word substring match.

- UDM_PARAM_MIN_WORD_LEN - defines minimal word length. Any word shorter this limit is considered to be a stopword. Please note that this parameter value is inclusive, i.e. if UDM_PARAM_MIN_WORD_LEN=3, a word 3 characters long will not be considered a stopword, while a word 2 characters long will be. Default value is 1.

- UDM_PARAM_ISPELL_PREFIXES - Possible values: UDM_PREFIXES_ENABLED and UDM_PREFIXES_DISABLED, that respectively enable or disable using prefixes. E.g. if a word "tested" is in search query, also words like "test", "testing", etc. Only suffixes are supported by default. Prefixes usually change word meanings, for example if somebody is searching for the word "tested" one hardly wants "untested" to be found. Prefixes support may also be found useful for site's spelling checking purposes. In order to enable ispell, you have to load ispell data with **udm_load_ispell_data()**.

- UDM_PARAM_CROSS_WORDS - enables or disables crosswords support. Possible values: UDM_CROSS_WORDS_ENABLED and UDM_CROSS_WORDS_DISABLED.

  The crosswords feature allows to assign words between and also to a document this link leads to. It works in SQL database mode and is not supported in built-in database and Cachemode.

  Nota: Crosswords are supported only in mnoGoSearch 3.1.11 or later.

- UDM_PARAM_VARDIR - specifies a custom path to directory where indexer stores data when using built-in database and in cache mode. By default */var* directory of mnoGoSearch installation is used. Can have only string values. The parameter is available in PHP 4.1.0 or later.

LXXV. Mohawk Software Session Handler Functions

Introducción

msession is an interface to a high speed session daemon which can run either locally or remotely. It is designed to provide consistent session management for a PHP web farm. More Information about msession and the session server software itself can be found at http://devel.mohawksoft.com/msession.html.

> **Nota:** Esta extensión no está disponible en plataformas Windows

---

Requirimientos

Instalación

To enable Msession support configure PHP *--with-msession[=DIR]*, where DIR is the Msession install directory.

Configuración en tiempo de ejecución

Tipos de recursos

Constantes predefinidas

Tabla de contenidos

msession_connect

(PHP 4 >= 4.2.0, PHP 5)

msession_connect -- Connect to msession server

Description

bool **msession_connect** (string host, string port)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

msession_count

(PHP 4 >= 4.2.0, PHP 5)

msession_count -- Get session count

Description

int **msession_count** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

msession_create

(PHP 4 >= 4.2.0, PHP 5)

msession_create -- Create a session

Description

bool **msession_create** (string session)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

msession_destroy

(PHP 4 >= 4.2.0, PHP 5)

msession_destroy -- Destroy a session

Description

bool **msession_destroy** (string name)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

msession_disconnect

(PHP 4 >= 4.2.0, PHP 5)

msession_disconnect -- Close connection to msession server

Description

void **msession_disconnect** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

msession_find

(PHP 4 >= 4.2.0, PHP 5)

msession_find -- Find all sessions with name and value

Description

array **msession_find** (string name, string value)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

msession_get_array

(PHP 4 >= 4.2.0, PHP 5)

msession_get_array -- Get array of msession variables

Description

array **msession_get_array** (string session)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

msession_get_data

(PHP 4 >= 4.2.0, PHP 5)

msession_get_data -- Get data session unstructured data

Description

string **msession_get_data** (string session)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

msession_get

(PHP 4 >= 4.2.0, PHP 5)

msession_get -- Get value from session

Description

string **msession_get** (string session, string name, string value)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

msession_inc

(PHP 4 >= 4.2.0, PHP 5)

msession_inc -- Increment value in session

Description

string **msession_inc** (string session, string name)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

msession_list

(PHP 4 >= 4.2.0, PHP 5)

msession_list -- List all sessions

Description

array **msession_list** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

msession_listvar

(PHP 4 >= 4.2.0, PHP 5)

msession_listvar -- List sessions with variable

Description

array **msession_listvar** (string name)

Returns an associative array of value/session for all sessions with a variable named *name*.

Used for searching sessions with common attributes.

msession_lock

(PHP 4 >= 4.2.0, PHP 5)

msession_lock -- Lock a session

Description

int **msession_lock** (string name)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

msession_plugin

(PHP 4 >= 4.2.0, PHP 5)

msession_plugin -- Call an escape function within the msession personality plugin

Description

string **msession_plugin** (string session, string val [, string param])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

msession_randstr

(PHP 4 >= 4.2.0, PHP 5)

msession_randstr -- Get random string

Description

string **msession_randstr** (int param)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

msession_set_array

(PHP 4 >= 4.2.0, PHP 5)

msession_set_array -- Set msession variables from an array

Description

bool **msession_set_array** (string session, array tuples)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

msession_set_data

(PHP 4 >= 4.2.0, PHP 5)

msession_set_data -- Set data session unstructured data

Description

bool **msession_set_data** (string session, string value)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

msession_set

(PHP 4 >= 4.2.0, PHP 5)

msession_set -- Set value in session

Description

bool **msession_set** (string session, string name, string value)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

msession_timeout

(PHP 4 >= 4.2.0, PHP 5)

msession_timeout -- Set/get session timeout

Description

int **msession_timeout** (string session [, int param])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

msession_uniq

(PHP 4 >= 4.2.0, PHP 5)

msession_uniq -- Get unique id

Description

string **msession_uniq** (int param)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

msession_unlock

(PHP 4 >= 4.2.0, PHP 5)

msession_unlock -- Unlock a session

Description

int **msession_unlock** (string session, int key)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

LXXVI. Funciones mSQL

Introducción

Estas funciones le permiten acceder a servidores de bases de datos mSQL. Puede encontrar más información sobre mSQL en http://www.hughes.com.au/.

Instalación

In order to have these functions available, you must compile PHP with msql support by using the *--with-msql[=DIR]* option. DIR is the mSQL base install directory, defaults to `/usr/local/msql3`.

> **Note to Win32 Users:** In order to enable this module on a Windows environment, you must copy `msql.dll` from the DLL folder of the PHP/Win32 binary package to the SYSTEM32 folder of your windows machine. (Ex: `C:\WINNT\SYSTEM32` or `C:\WINDOWS\SYSTEM32`)

Configuración en tiempo de ejecución

El comportamiento de estas funciones está afectado por los valores definidos en `php.ini`.

Tabla 1. mSQL configuration options

Name	Default	Changeable
msql.allow_persistent	"On"	PHP_INI_SYSTEM
msql.max_persistent	"-1"	PHP_INI_SYSTEM
msql.max_links	"-1"	PHP_INI_SYSTEM

For further details and definitions of the PHP_INI_* constants, see the Apéndice H.

A continuación se presenta una corta explicación de las directivas de configuración.

msql.allow_persistent **boolean**

Whether to allow persistent mSQL connections.

msql.max_persistent **integer**

The maximum number of persistent mSQL connections per process.

msql.max_links **integer**

The maximum number of mSQL connections per process, including persistent connections.

Tipos de recursos

Existen dos tipos de recurso usados en el módulo mSQL. El primero es el identificador de enlace para una conexión de base de datos, el segundo es un recurso que contiene el resultado de una consulta.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

`MSQL_ASSOC` (**integer**)

`MSQL_NUM` (**integer**)

`MSQL_BOTH` (**integer**)

Ejemplos

Este sencillo ejemplo muestra el modo de crear una conexión, ejecutar una consulta, imprimir las filas de resultado y desconectarse de una base de datos mSQL.

Ejemplo 1. Ejemplo de uso de mSQL

```
<?php
/* Conexion, seleccion de una base de datos */
$enlace = msql_connect('localhost', 'nombre_usuario', 'contrasenya')
    or die('No pudo crear una conexi&oacute;n: ' . msql_error($enlace));

msql_select_db('base_de_datos', $enlace)
    or die('No pudo seleccionarse la base de datos');

/* Realizar una consulta SQL */
$consulta = 'SELECT * FROM mi_tabla';
$resultado = msql_query($consulta, $enlace) or die('La consulta fall&oacute;: ' . msql_

/* Impresion de resultados en HTML */
echo "<table>\n";
while ($fila = msql_fetch_array($resultado, MSQL_ASSOC)) {
    echo "\t<tr>\n";
    foreach ($fila as $valor_col) {
        echo "\t\t<td>$valor_col</td>\n";
    }
    echo "\t</tr>\n";
}
echo "</table>\n";

/* Liberar el conjunto de resultados */
msql_free_result($resultado);

/* Cerrar la conexion */
msql_close($enlace);
?>
```

Tabla de contenidos

msql_free_result -- libera la memoria del resultado
msql_list_dbs -- lista las bases de datos mSQL en el servidor
msql_list_fields -- lista los campos del resultado
msql_list_tables -- lista las tablas de una base de datos mSQL
msql_num_fields -- obtiene el número de campos de un resultado
msql_num_rows -- obtiene el número de filas de un resultado
msql_numfields -- obtiene el número de campos de un resultado
msql_numrows -- obtiene el número de filas en el resultado
msql_pconnect -- abre una conexión mSQL persistente
msql_query -- envía una consulta mSQL
msql_regcase -- construye una expresión regular para una búsqueda que no distinga mayúsculas/minúsculas
msql_result -- obtiene datos resultado
msql_select_db -- selecciona una base de datos mSQL
msql_tablename -- obtiene el nombre de la tabla de un campo
msql -- ejecuta una consulta mSQL

msql_affected_rows

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

msql_affected_rows -- devuelve el número de filas involucradas

Descripción

int **msql_affected_rows** (int query_identifier)

Devuelve el número de filas involucradas ("tocadas") por una consulta específica (i.e. el número de filas devueltas por una SELECT, el número de filas modificadas por una actualización (update), o el número de filas suprimidas por una eliminación (delete)).

Véase también: **msql_query()**

msql_close

(PHP 3, PHP 4 , PHP 5)

msql_close -- cierra una conexión mSQL

Descripción

int **msql_close** (int link_identifier)

Devuelve **TRUE** si tiene éxito, **FALSE** en caso de error.

msql_close() cierra la conexión con una base de datos mSQL que está asociada con el identificador de conexión (link identifier) especificado. Si no se especifica el identificador de conexión, se asume la última conexión abierta.

Advierta que ésto no es necesario habitualmente, las conexiones abiertas no-persistentes son cerradas automáticamente a la conclusión de la ejecución del script.

msql_close() no cerrará las conexiones permanentes creadas por **msql_pconnect()**.

Véase también: **msql_connect()** y **msql_pconnect()**.

msql_connect

(PHP 3, PHP 4 , PHP 5)

msql_connect -- abre una conexión mSQL

Descripción

int **msql_connect** (string hostname)

Devuelve un identificador de conexión mSQL positivo si tiene éxito, o **FALSE** en caso de error.

msql_connect() establece una conexión con un servidor mSQL. El argumento hostname es opcional, y en caso de que falte, se asume localhost.

En caso de que se haga una segunda llamada a msql_connect() con los mismos argumentos, no se establece una nueva conexión, en lugar de eso, se devuelve el identificador de conexión ya abierto.

La conexión con el servidor se cerrará tan pronto como la ejecución del script finalice, a menos que sea cerrada antes explícitamente por una llamada a **msql_close()**.

Véase también: **msql_pconnect()**, **msql_close()**.

msql_create_db

(PHP 3, PHP 4 , PHP 5)

msql_create_db -- crea una base de datos mSQL

Descripción

int **msql_create_db** (string database name [, int link_identifier])

msql_create_db() intenta crear una base de datos nueva en el servidor asociado con el identificador de conexión (link identifier) especificado.

Véase también: **msql_drop_db()**.

msql_createdb

(PHP 3, PHP 4 , PHP 5)

msql_createdb -- crea una base de datos mSQL

Descripción

int **msql_createdb** (string database name [, int link_identifier])

Idéntica a **msql_create_db()**.

msql_data_seek

(PHP 3, PHP 4 , PHP 5)

msql_data_seek -- desplaza el puntero interno de fila

Descripción

int **msql_data_seek** (int query_identifier, int row_number)

Devuelve **TRUE** si tiene éxito, **FALSE** en caso de fallo.

msql_data_seek() desplaza el puntero interno de fila del resultado mSQL asociado con el identificador de consulta (query identifier) especificado para que apunte al número de fila (row number) proporcionado. La llamada posterior a **msql_fetch_row()** devolverá esa fila.

Véase también: **msql_fetch_row()**.

msql_db_query

(PHP 3, PHP 4 , PHP 5)

msql_db_query -- Send mSQL query

Description

resource **msql_db_query** (string database, string query [, resource link_identifier])

Returns a positive mSQL query identifier to the query result, or **FALSE** on error.

msql_db_query() selects a database and executes a query on it. If the optional *link_identifier* is not specified, the function will try to find an open link to the mSQL server; if no such link is found it will try to create one as if **msql_connect()** was called with no arguments.

See also **msql_connect()**.

msql_dbname

(PHP 3, PHP 4 , PHP 5)

msql_dbname -- obtiene el nombre de la base de datos mSQL actual

Descripción

string **msql_dbname** (int query_identifier, int i)

msql_dbname() devuelve el nombre de base de datos almacenado en la posición *i* del puntero devuelto desde la función **msql_listdbs()**. La función **msql_numrows()** puede utilizarse para determinar cuantos nombres de base de datos hay disponibles.

msql_drop_db

(PHP 3, PHP 4 , PHP 5)

msql_drop_db -- elimina (suprime) una base de datos mSQL

Descripción

int **msql_drop_db** (string database_name, int link_identifier)

Devuelve **TRUE** si tiene éxito, **FALSE** en caso de fallo.

msql_drop_db() intenta eliminar (suprimir) una base de datos completa del servidor asociado con el identificador de conexión (link identifier) especificado.

Véase también: **msql_create_db()**.

msql_error

(PHP 3, PHP 4 , PHP 5)

msql_error -- devuelve el mensaje de error de la última llamada msql

Descripción

string **msql_error** ( )

Los errores que devuelve el servidor de base de datos mSQL no dan mucha información sobre el problema. Por este motivo, utilice estas funciones para recuperar la cadena de carácteres del error.

msql_fetch_array

(PHP 3, PHP 4 , PHP 5)

msql_fetch_array -- recupera una fila como un array

Descripción

int **msql_fetch_array** (int query_identifier [, int result_type])

Devuelve un array que se corresponde con la fila recuperada, o **FALSE** si no hay más filas.

msql_fetch_array() es una versión ampliada de **msql_fetch_row()**. Además de almacenar los datos en los índices numéricos del array resultado, también almacena los datos en índices asociativos, utilizando los nombres de los campos como claves.

El segundo argumento opcional *result_type* en **msql_fetch_array()** es una constante y puede tomar los valores siguientes: MSQL_ASSOC, MSQL_NUM, y MYSQL_BOTH.

Sea precavido si está recuperando resultados de una consulta que puede devolver un registro que contiene un único campo que tiene un valor de 0 (o una cadena de carácteres vacía, o **NULL**).

Un aspecto importante a tener en cuenta es que el uso de **msql_fetch_array()** NO es significativamente más lento que el uso de **msql_fetch_row()**, mientras que proporciona un valor añadido significativo.

Para detalles adicionales, véase también **msql_fetch_row()**

msql_fetch_field

(PHP 3, PHP 4 , PHP 5)

msql_fetch_field -- obtiene información de campo

Descripción

object **msql_fetch_field** (int query_identifier, int field_offset)

Devuelve un objeto que contiene la información del campo

msql_fetch_field() puede utilizarse para obtener información sobre campos del resultado de una consulta. Si no se especifica el desplazamiento del campo, se recupera el campo siguiente que no haya sido aún recuperado por msql_fetch_field().

Las propiedades del objeto son:

- name - nombre de la columna
- table - nombre de la tabla a la que pertenece la columna
- not_null - 1 si la columna no puede ser nula
- primary_key - 1 si la columna es una clave primaria
- unique - 1 si la columna es una clave única
- type - tipo de la columna

Véase también **msql_field_seek()**.

msql_fetch_object

(PHP 3, PHP 4 , PHP 5)

msql_fetch_object -- recupera una fila como un objeto

Descripción

int **msql_fetch_object** (int query_identifier [, int result_type])

Devuelve un objeto con las propiedades que corresponden a la fila recuperada, o **FALSE** si no hay más filas.

msql_fetch_object() es similar a **msql_fetch_array()**, con una diferencia - se devuelve un objeto en vez de un array. Indirectamente esto significa que sólo tiene acceso a los datos por los nombres de los campos, y no por sus desplazamientos (los números son nombres de propiedad no válidos).

El segundo parámetro opcional *result_type* en **msql_fetch_array()** es una constante y puede tomar los valores siguientes: MSQL_ASSOC, MSQL_NUM, y MSQL_BOTH.

Resumiendo, la función es indéntica a **msql_fetch_array()**, y casi tan rápida como **msql_fetch_row()** (la diferencia es insignificante).

Véase también: **msql_fetch_array()** y **msql_fetch_row()**.

msql_fetch_row

(PHP 3, PHP 4 , PHP 5)

msql_fetch_row -- obtiene una fila como un array enumerado

Descripción

array **msql_fetch_row** (int query_identifier)

Devuelve un array que se corresponde con la fila recuperada, o **FALSE** si no hay más filas.

msql_fetch_row() recupera una fila de datos del resultado asociado con el identificador de consulta (query identifier) especificado. La fila se devuelve en un array. Cada columna devuelta se almacena en un desplazamiento del array, comenzando en el desplazamiento 0.

Una llamada posterior a msql_fetch_row() devería devolver la fila siguiente del conjunto resultado, o **FALSE** si no hay más filas.

Véase también: **msql_fetch_array()**, **msql_fetch_object()**, **msql_data_seek()**, y **msql_result()**.

msql_field_flags

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

msql_field_flags -- Get field flags

Description

string **msql_field_flags** (resource query_identifier, int field_offset)

msql_field_flags() returns the field flags of the specified field. Currently this is either, "not **NULL**", "primary key", a combination of the two or "" (an empty string).

msql_field_len

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

msql_field_len -- Get field length

Description

int **msql_field_len** (resource query_identifier, int field_offset)

msql_field_len() returns the length of the specified field or **FALSE** on error.

msql_field_name

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

msql_field_name -- Get field name

Description

string **msql_field_name** (resource query_identifier, int field)

msql_field_name() returns the name of the specified *field* from the result resource *query_identifier*. *msql_field_name($result, 2);* will return the name of the second field in the result set associated with the result identifier.

msql_field_seek

(PHP 3, PHP 4 , PHP 5)

msql_field_seek -- establece el desplazamiento del campo

Descripción

int **msql_field_seek** (int query_identifier, int field_offset)

Se posiciona en el desplazamiento de campo (field offset) especificado. Si la siguiente llamada a **msql_fetch_field()** no incluye un desplazamiento de campo, este campo será el que se devuelva.

Véase también: **msql_fetch_field()**.

msql_field_table

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

msql_field_table -- Get table name for field

Description

int **msql_field_table** (int query_identifier, int field)

Returns the name of the table *field* was fetched from.

This function returns **FALSE** on failure.

msql_field_type

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

msql_field_type -- Get field type

Description

string **msql_field_type** (resource query_identifier, int field_offset)

msql_field_type() is similar to the **msql_field_name()** function. The arguments are identical, but the field type is returned. This will be one of "int", "char" or "real".

This function returns **FALSE** on failure.

msql_fieldflags

(PHP 3, PHP 4 , PHP 5)

msql_fieldflags -- obtiene los flags del campo

Descripción

string **msql_fieldflags** (int query_identifier, int i)

msql_fieldflags() obtiene los flags del campo (field) especificado. Actualmente pueden ser, "not **NULL**", "primary key", una combinación de ambos, o "" (cadena vacía).

msql_fieldlen

(PHP 3, PHP 4 , PHP 5)

msql_fieldlen -- obtiene la longitud del campo

Descripción

int **msql_fieldlen** (int query_identifier, int i)

msql_fieldlen() devuelve la longitud del campo especificado.

msql_fieldname

(PHP 3, PHP 4 , PHP 5)

msql_fieldname -- obtiene el nombre del campo

Descripción

string **msql_fieldname** (int query_identifier, int field)

msql_fieldname() devuelve el nombre del campo especificado. *query_identifier* es el identificador de consulta, y *field* es el índice del campo. *msql_fieldname($result, 2);* devolverá el nombre del segundo campo del resultado asociado con el identificador result.

msql_fieldtable

(PHP 3, PHP 4 , PHP 5)

msql_fieldtable -- obtiene el nombre de la tabla de un campo

Descripción

int **msql_fieldtable** (int query_identifier, int field)

Devuelve el nombre de la tabla desde la que se ha recuperado el campo (*field*)

msql_fieldtype

(PHP 3, PHP 4 , PHP 5)

msql_fieldtype -- obtiene el tipo del campo

Descripción

string **msql_fieldtype** (int query_identifier, int i)

msql_fieldtype() es similar a la función **msql_fieldname()**. Los argumentos son idénticos, pero se devuelve el tipo del campo. Este será "int", "char" o "real".

msql_free_result

(PHP 3, PHP 4 , PHP 5)

msql_free_result -- libera la memoria del resultado

Descripción

int **msql_free_result** (int query_identifier)

msql_free_result() libera la memoria asociada con *query_identifier*. Cuando PHP completa una petición, esta memoria es liberada automáticamente, por este motivo solo es necesario llamar a esta función cuando se desea estar seguro de que no se utiliza demasiada memoria mientras se está ejecutando el script.

msql_list_dbs

(PHP 3, PHP 4 , PHP 5)

msql_list_dbs -- lista las bases de datos mSQL en el servidor

Descripción

int **msql_list_dbs** (void)

msql_list_dbs() devolverá un puntero al resultado que contiene las bases de datos disponibles desde el daemon msql en uso. Utilice la función **msql_dbname()** para recorrer este puntero.

msql_list_fields

(PHP 3, PHP 4 , PHP 5)

msql_list_fields -- lista los campos del resultado

Descripción

int **msql_list_fields** (string database, string tablename)

msql_list_fields() recupera información sobre el nombre de tabla (tablename) dado. Los argumentos son el nombre de la base de datos (database name) y el nombre de la tabla (table name). Se devuelve un puntero al resultado que puede utilizarse con **msql_fieldflags()**, **msql_fieldlen()**, **msql_fieldname()**, y **msql_fieldtype()**. Un indetificador de consulta (query identifier) es un entero positivo. La función devuelve *-1* si ocurre un error. En *$phperrmsg* se almacena una cadena de carácteres describiendo el error, y a menos que la función sea llamada como *@msql_list_fields()* esta cadena de error puede ser impresa.

Véase también **msql_error()**.

msql_list_tables

(PHP 3, PHP 4 , PHP 5)

msql_list_tables -- lista las tablas de una base de datos mSQL

Descripción

int **msql_list_tables** (string database)

msql_list_tables() toma un nombre de base de datos y devuelve un puntero similar al de la función **msql()** La función **msql_tablename()** debería utilizarse para obtener los nombres reales de las tablas del puntero devuelto.

msql_num_fields

(PHP 3, PHP 4 , PHP 5)

msql_num_fields -- obtiene el número de campos de un resultado

Descripción

int **msql_num_fields** (int query_identifier)

msql_num_fields() devuelve el número de campos de un conjunto resultado.

Véase también: **msql()**, **msql_query()**, **msql_fetch_field()**, y **msql_num_rows()**.

msql_num_rows

(PHP 3, PHP 4 , PHP 5)

msql_num_rows -- obtiene el número de filas de un resultado

Descripción

int **msql_num_rows** (int query_identifier)

msql_num_rows() devuelve el número de filas de un conjunto resultado.

Véase también: **msql()**, **msql_query()**, y **msql_fetch_row()**.

msql_numfields

(PHP 3, PHP 4 , PHP 5)

msql_numfields -- obtiene el número de campos de un resultado

Descripción

int **msql_numfields** (int query_identifier)

Idéntica a **msql_num_fields()**.

msql_numrows

(PHP 3, PHP 4 , PHP 5)

msql_numrows -- obtiene el número de filas en el resultado

Descripción

int **msql_numrows** (void)

Idéntica a **msql_num_rows()**.

msql_pconnect

(PHP 3, PHP 4 , PHP 5)

msql_pconnect -- abre una conexión mSQL persistente

Descripción

int **msql_pconnect** (string hostname)

En caso de éxito devuelve un identificador de conexión mSQL persistente positivo, o **FALSE** en caso de error.

msql_pconnect() se comporta de forma similar a **msql_connect()** con dos diferencias importantes.

Primero, cuando se conecta, la función debe intentar primero localizar una conexión (persistente) que ya esté abierta en el mismo host. Si se encuentra uno, se devuelve un identificador para el mismo en vez de abrir una conexión nueva.

Segundo, la conexión con el servidor SQL no se cerrará cuando la ejecución del script finalice. Al contrario, la conexión permanecerá abierta para un uso futuro (**msql_close()** no cerrará las conexiones abiertas por msql_pconnect()).

Este tipo de conexiones son por ello denominadas 'persistentes'.

msql_query

(PHP 3, PHP 4 , PHP 5)

msql_query -- envía una consulta mSQL

Descripción

int **msql_query** (string query, int link_identifier)

msql_query() envía una consulta a la base de datos activa actual en el servidor que está asociada con el identificador de conexión (link identifier) especificado. Si no se especifica el identificador de

conexión, se asume la última conexión abierta. Si no hay ninguna conexión abierta, la función intenta establecer una conexión como si se hubiera llamado a **msql_connect()**, y la utiliza.

En caso de éxito devuelve un identificador de consulta mSQL positivo, o **FALSE** en caso de error.

Véase también: **msql()**, **msql_select_db()**, y **msql_connect()**.

msql_regcase

msql_regcase -- construye una expresión regular para una búsqueda que no distinga mayúsculas/minúsculas

Descripción

Véase **sql_regcase()**.

msql_result

(PHP 3, PHP 4 , PHP 5)

msql_result -- obtiene datos resultado

Descripción

int **msql_result** (int query_identifier, int i, mixed field)

Devuelve el contenido de la celda en la fila y desplazamiento del conjunto resultado mSQL especificado.

msql_result() devuelve el contenido de una celda de un conjunto resultado mSQL. El argumento campo (field) puede ser el desplazamiento del campo, el nombre del campo, o el nombre de la tabla punto nombre del campo (nombretabla.nombrecampo). Si el nombre de la columna tiene un alias ('select foo as bar from...'), utilice el alias en vez del nombre de la columna.

Cuando se trabaja con conjuntos de resultados grandes, debería considerar el uso de las funciones que recuperen filas completas (especificadas más abajo). Como estas funciones recuperan el contenido de varias celdas en una única llamada de función, son MUCHO más rápidas que msql_result(). Advierta también que especificar un desplazamiento numérico para el argumento campo (field) es mucho más rápido que especificar un argumento nombrecampo o nombretabla.nombrecampo.

Alternativas de alto-rendimiento recomendadas: **msql_fetch_row()**, **msql_fetch_array()**, y **msql_fetch_object()**.

msql_select_db

(PHP 3, PHP 4 , PHP 5)

msql_select_db -- selecciona una base de datos mSQL

Descripción

int **msql_select_db** (string database_name, int link_identifier)

Devuelve **TRUE** si tiene éxito, **FALSE** en caso contrario.

msql_select_db() establece la base de datos activa actual en el servidor que está asociada con el identificador de conexión (link identifier) suministrado. Si no se especifica el identificador de conexión, se asumne la última conexión abierta. Si no hay ninguna conexión abierta la función intentará establecer una conexión como si se hubiera llamado a sql_connect(), y la utiliza.

Cada llamada posterior a **msql_query()** se hará en la base de datos activa.

Véase también: **msql_connect()**, **msql_pconnect()**, y **msql_query()**.

msql_tablename

(PHP 3, PHP 4 , PHP 5)

msql_tablename -- obtiene el nombre de la tabla de un campo

Descripción

string **msql_tablename** (int query_identifier, int field)

msql_tablename() toma un puntero resultado devuelto por la función **msql_list_tables()** como un índice entero y devuelve el nombre de una tabla. La función **msql_numrows()** puede utilizarse para determinar el número de tablas del puntero resultado.

Ejemplo 1. msql_tablename() example

```
<?php
msql_connect ("localhost");
$result = msql_list_tables("wisconsin");
$i = 0;
while ($i < msql_numrows($result)) {
    $tb_names[$i] = msql_tablename($result, $i);
    echo $tb_names[$i] . "<BR>";
    $i++;
}
?>
```

msql

(PHP 3, PHP 4 , PHP 5)

msql -- ejecuta una consulta mSQL

Descripción

int **msql** (string database, string query, int link_identifier)

Devuelve un identificador de consulta mSQL positivo en el resultado de la consulta, o **FALSE** en caso de error.

msql() selecciona una base de datos y ejecuta una consulta en ella. Si no se especifica el identificador de conexión (link identifier), la función intentará encontrar una conexión abierta en el servidor mSQL y en el caso de que no se encontrase intentará crear uno como si se llamase a **msql_connect()** sin parámetros (véase **msql_connect()**).

LXXVII. Funciones de Microsoft SQL Server

Tabla de contenidos

mssql_bind

(PHP 4 >= 4.1.0, PHP 5)

mssql_bind -- Adds a parameter to a stored procedure or a remote stored procedure

Description

bool **mssql_bind** (resource stmt, string param_name, mixed &var, int type [, int is_output [, int is_null [, int maxlen]]])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

See also **mssql_execute()**, **mssql_free_statement()**, and **mssql_init()**.

mssql_close

(PHP 3, PHP 4 , PHP 5)

mssql_close -- cierra una conexión con MS SQL Server

Descripción

int **mssql_close** (int link_identifier)

Devuleve: **TRUE** si se finaliza con éxito, **FALSE** si se produce un error

mssql_close() cierra la conexión con una base de datos MS SQL Server que está asociada al identificador especificado. Si el identificador no se especifica, se asume la última conexión abierta.

Observe que normalmente esto no es necesario, ya que las conexiones no-persistentes abiertas se cierran automaticamente en cuanto finaliza el script.

mssql_close() no cerrará conexiones persistentes generadas por mssql_pconnect().

Ver también: **mssql_connect()**, **mssql_pconnect()**.

mssql_connect

(PHP 3, PHP 4 , PHP 5)

mssql_connect -- Abrir una conexión con MS SQL server

Descripción

int **mssql_connect** ([string nombre_servidor [, string nombre_usuario [, string contrasenya]]])

Devuelve: Un identificador de enlace MS SQL positivo de tener éxito, o **FALSE** si ocurre un error.

mssql_connect() establece una conexión con un servidor MS SQL. El argumento nombre_servidor debe ser un nombre de servidor válido, que esté definido en el archivo 'interfaces'.

En caso de que se realice una segunda llamada a **mssql_connect()** con los mismos argumentos, no se establecerá un nuevo enlace, sino que se devolverá el identificador de enlace de la conexión abierta anteriormente.

El enlace con el servidor será cerrado tan pronto como finalice la ejecución del script, a menos que se cierre antes mediante una llamada explícita a **mssql_close()**.

Vea también **mssql_pconnect()**, **mssql_close()**.

mssql_data_seek

(PHP 3, PHP 4 , PHP 5)

mssql_data_seek -- mueve el puntero interno de las filas

Descripción

int **mssql_data_seek** (int result_identifier, int row_number)

Devuelve: **`TRUE`** si se ejecuta con éxito, **`FALSE`** si falla.

mssql_data_seek() mueve el puntero interno de la consulta MS SQL asociada al result_identifier especificado, para que apunte al número de fila especificada. La siguiente llamada a **mssql_fetch_row()** devolverá esa fila.

Ver también: **mssql_data_seek()**.

mssql_execute

(PHP 4 >= 4.1.0, PHP 5)

mssql_execute -- Executes a stored procedure on a MS SQL server database

Description

mixed **mssql_execute** (resource stmt [, bool skip_results])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

> **Nota:** If the stored procedure returns parameters or a return value these will be available after the call to **mssql_execute()** unless the stored procedure returns more than one result set. In that case use **mssql_next_result()** to shift through the results. When the last result has been processed the output parameters and return values will be available.

See also **mssql_bind()**, **mssql_free_statement()**, and **mssql_init()**.

mssql_fetch_array

(PHP 3, PHP 4 , PHP 5)

mssql_fetch_array -- Captura la fila en un array

Descripción

int **mssql_fetch_array** (int result)

Devuelve: Un array que corresponde a la fila capturada, o **FALSE** si no hay más filas.

mssql_fetch_array() es una versión extendida de **mssql_fetch_row()**. AÃ‚Â¤ade el almacenar los datos en los índices numéricos del array resultante, también almacena los datos en índices asociativos, usando los nombres de los campos como claves.

Una observación a tener en cuenta es, que usar mssql_fetch_array() NO es más lento que usar mssql_fetch_row(), mientras que esta provee un valor aÃ‚Â¤adido significativo.

Para más detalles, ver también **mssql_fetch_row()**

mssql_fetch_assoc

(PHP 4 >= 4.2.0, PHP 5)

mssql_fetch_assoc -- Returns an associative array of the current row in the result set specified by result_id

Description

array **mssql_fetch_assoc** (resource result_id)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mssql_fetch_batch

(PHP 4 >= 4.0.4, PHP 5)

mssql_fetch_batch -- Returns the next batch of records

Description

int **mssql_fetch_batch** (resource result_index)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mssql_fetch_field

(PHP 3, PHP 4 , PHP 5)

mssql_fetch_field -- obtiene la información de los campos

Descripción

object **mssql_fetch_field** (int result, int field_offset)

Devuleve un objeto que contiene información de los campos.

mssql_fetch_field() se puede usar para obtener información acerca de los campos pertenecientes al resultado de una consulta. Si el parámetro field_offset no es especificado, se devuelve la información del siguiente campo que todavía no ha sido devuelto por mssql_fetch_field().

Las propiedades de este objeto son:

- name - nombre de la columna. si la columna es el resultado de una función, esta propiedad vale #N, donde #N es un número de serie.
- column_source - la tabla de donde se tomó la columna
- max_length - longitud máxima de columna
- numeric - 1 si la columna es numérica

Ver también **mssql_field_seek()**

mssql_fetch_object

(PHP 3, PHP 5)

mssql_fetch_object -- captura la fila como un objeto

Descripción

int **mssql_fetch_object** (int result)

Devuelve: Un objeto con propiedades que se corresponden con la fila capturada, o **FALSE** si no hay más filas.

mssql_fetch_object() es parecida a **mssql_fetch_array()**, con una diferencia - devuelve un objeto en vez de un array. Indirectamente, esto significa que sólo se puede acceder a los datos por el nombre de los campos, y no por sus posiciones en el objeto (los números no son nombres de propiedades válidas).

La función es idéntica a **mssql_fetch_array()**, y casi tan rápida como **mssql_fetch_row()** (la diferencia es insignificante).

Ver también: **mssql_fetch-array()** and **mssql_fetch-row()**.

mssql_fetch_row

(PHP 3, PHP 4 , PHP 5)

mssql_fetch_row -- obtiene la fila como un array numerado

Descripción

array **mssql_fetch_row** (int result)

Devuleve: Un array que corresponde a la fila capturada, o **FALSE** si no hay más filas.

mssql_fetch_row() captura una fila de datos pertenecientes al resultado asociado con el identificador de resultado especificado. La fila es devuelta como un array. Cada columna de resultados es almacenada en una posición del array, comenzando en la posición 0.

Siguientes llamadas a mssql_fetch_rows() devolverían las filas siguientes del result set, o **FALSE** si no hay mas filas.

Ver también: **mssql_fetch_array()**, **mssql_fetch_object()**, **mssql_data_seek()**, **mssql_fetch_lengths()**, and **mssql_result()**.

mssql_field_length

(PHP 3>= 3.0.3, PHP 4 , PHP 5)

mssql_field_length -- Get the length of a field

Description

int **mssql_field_length** (resource result [, int offset])

This function returns the length of field no. *offset* in result *result*. If *offset* is omitted, the current field is used.

> **Note to Win32 Users:** Due to a limitation in the underlying API used by PHP (MS DbLib C API), the length of VARCHAR fields is limited to *255*. If you need to store more data, use a TEXT field instead.

mssql_field_name

(PHP 3>= 3.0.3, PHP 4 , PHP 5)

mssql_field_name -- Get the name of a field

Description

string **mssql_field_name** (resource result [, int offset])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mssql_field_seek

(PHP 3, PHP 4 , PHP 5)

mssql_field_seek -- set field offset

Descripción

int **mssql_field_seek** (int result, int field_offset)

Se posiciona en el campo especificado por el parámetro field_offset. Si la siguiente llamada a **mssql_fetch_field()** no incluye el parámetro field_offset, lo que devuelve la función es el campo.

Ver también: **mssql_fetch_field()**.

mssql_field_type

(PHP 3>= 3.0.3, PHP 4 , PHP 5)

mssql_field_type -- Gets the type of a field

Description

string **mssql_field_type** (resource result [, int offset])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mssql_free_result

(PHP 3, PHP 4 , PHP 5)

mssql_free_result -- libera de la memoria el resultado de una consulta

Descripción

int **mssql_free_result** (int result)

mssql_free_result() sólo se necesita llamarla si le preocupa el estar usando mucha memoria mientras se está ejecutando el script. Toda el resultado en memoria será liberado automaticamente cuando finalice el script, puede llamar a **mssql_free_result()** con el identificador de la consulta como argumento y la consulta asociada será liberada de la memoria.

mssql_free_statement

(PHP 4 >= 4.3.2, PHP 5)

mssql_free_statement -- Free statement memory

Description

bool **mssql_free_statement** (resource statement)

mssql_free_statement() only needs to be called if you are worried about using too much memory while your script is running. All statement memory will automatically be freed when the script ends. You may call **mssql_free_statement()** with the statement identifier as an argument and the associated statement memory will be freed.

See also **mssql_bind()**, **mssql_execute()**, and **mssql_init()**

mssql_get_last_message

(PHP 3, PHP 4 , PHP 5)

mssql_get_last_message -- Returns the last message from the server

Description

string **mssql_get_last_message** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mssql_guid_string

(PHP 4 >= 4.1.0, PHP 5)

mssql_guid_string -- Converts a 16 byte binary GUID to a string

Description

string **mssql_guid_string** (string binary [, int short_format])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mssql_init

(PHP 4 >= 4.1.0, PHP 5)

mssql_init -- Initializes a stored procedure or a remote stored procedure

Description

int **mssql_init** (string sp_name [, resource conn_id])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

See also **mssql_bind()**, **mssql_execute()**, and **mssql_free_statement()**

mssql_min_error_severity

(PHP 3, PHP 4 , PHP 5)

mssql_min_error_severity -- Sets the lower error severity

Description

void **mssql_min_error_severity** (int severity)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mssql_min_message_severity

(PHP 3, PHP 4 , PHP 5)

mssql_min_message_severity -- Sets the lower message severity

Description

void **mssql_min_message_severity** (int severity)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mssql_next_result

(PHP 4 >= 4.0.5, PHP 5)

mssql_next_result -- Move the internal result pointer to the next result

Description

bool **mssql_next_result** (resource result_id)

When sending more than one SQL statement to the server or executing a stored procedure with multiple results, it will cause the server to return multiple result sets. This function will test for additional results available form the server. If an additional result set exists it will free the existing result set and prepare to fetch the rows from the new result set. The function will return **TRUE** if an additional result set was available or **FALSE** otherwise.

Ejemplo 1. mssql_next_result() example

```
<?php
    $link = mssql_connect("localhost", "userid", "secret");
    mssql_select_db("MyDB", $link);
    $SQL = "Select * from table1 select * from table2";
    $rs = mssql_query($SQL, $link);
    do {
        while ($row = mssql_fetch_row($rs)) {
        }
    } while (mssql_next_result($rs));
    mssql_free_result($rs);
    mssql_close($link);
?>
```

mssql_num_fields

(PHP 3, PHP 4 , PHP 5)

mssql_num_fields -- obtiene el número de campos de la consulta

Descripción

int **mssql_num_fields** (int result)

mssql_num_fields() devuelve el número de campos de la consulta o result set.

Ver también: **mssql_db_query()**, **mssql_query()**, **mssql_fetch_field()**, **mssql_num_rows()**.

mssql_num_rows

(PHP 3, PHP 4 , PHP 5)

mssql_num_rows -- obtiene el número de filas de la consulta

Descripción

int **mssql_num_rows** (string result)

mssql_num_rows() devuelve el número de filas de la consulta o result set.

Ver también: **mssql_db_query()**, **mssql_query()** and, **mssql_fetch_row()**.

mssql_pconnect

(PHP 3, PHP 4 , PHP 5)

mssql_pconnect -- abre una conexión persistenete con MS SQL

Descripción

int **mssql_pconnect** (string servername, string username, string password)

Devuelve: Un identificador persistente postivo si no hay error, o **`FALSE`** si se produce alguno

mssql_pconnect() funciona de la misma forma que **mssql_connect()** aunque con dos grandes diferencias.

La primera es que cuando intenta conectar, la función intentará encontrar un enlace (persistente) que ya esté abierto en el mismo ordenador, nombre de usuario y contraseÃ‚Â¤a. Si lo encuentra, la funcion devolverá el identificador de esta en vez de abrir una nueva conexión.

Y la segunda, la conexión con el servidor no se cerrará cuando finalice la ejecución del script. En vez de esto, el enlace permanecerá abierto para un uso futuro. (**mssql_close()** no cerrará enlaces establecidos por *mssql_pconnect()*).

Por consiguiente, este tipo de enlace es llamado 'persistente'.

mssql_query

(PHP 3, PHP 4 , PHP 5)

mssql_query -- envia una consulta MS SQL

Descripción

int **mssql_query** (string query, int link_identifier)

Devuelve: Un identificado de resultado valido si no hay error, o **`FALSE`** en caso contrario.

mssql_query() envia una petición de consulta a la base de datos activa en el servidor asociada al identificador de enlace especificado. Si el identificador del enlace no es especificado, se asume como abierto el último enlace. Si no hay ningún enlace abierto, la función intenta establecer un enlace como si **mssql_connect()** hubiera sido llamada, y lo usa.

Ver también: **mssql_db_query()**, **mssql_select_db()**, and **mssql_connect()**.

mssql_result

(PHP 3, PHP 4 , PHP 5)

mssql_result -- get result data

Descripción

int **mssql_result** (int result, int i, mixed field)

Devuelve: El contenido de la celda en la fila y posición del result set especificado.

mssql_result() devuelve el contenido de una celda del result set. El parametro field puede ser la posición del campo, o el nombre del campo o bién nombretabla.nombrecampo. Si el nombre de la columna ha sido renombrado ('select foo as bar from...'), use el alias en vez del nombre de la columna.

Trabajando con result sets de gran tamaÃ‚Â¤o, debería considerar el uso de una de las funciones que capturan una fila completa (especificadas abajo). Como estas funciones devuelven el contenido de múltiples celdas en una sóla llamada, estas son MUCHO más rápidas que mssql_result(). También, observe que especificar una posición númerica para el argumento field es mucho mas rápido que especificar el nombre de un campo o utilizar la forma nombretabla.nombrecampo como argumento.

Alternativas recomendadas para mayor rendimiento : **mssql_fetch_row()**, **mssql_fetch_array()**, y **mssql_fetch_object()**.

mssql_rows_affected

(PHP 4 >= 4.0.4, PHP 5)

mssql_rows_affected -- Returns the number of records affected by the query

Description

int **mssql_rows_affected** (resource conn_id)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

mssql_select_db

(PHP 3, PHP 4 , PHP 5)

mssql_select_db -- selecciona una base de datos MS SQL

Descripción

int **mssql_select_db** (string database_name, int link_identifier)

Devuelve: **`TRUE`** si todo va bién, **`FALSE`** si se produce un error

mssql_select_db() selecciona como base de datos activa del servidor, la que está asociada al identificador de enlace especificado. Si no se especifica nigún identificador, se asume el último enlace. Si no hay ningún enlace abierto, la función intentará establecer un enlace como si se llamara a la función **mssql_connect()**, y lo usa.

Cada llamada a **mssql_query()** será realizada sobre la base de datos activa.

Ver también: **mssql_connect()**, **mssql_pconnect()**, y **mssql_query()**

LXXVIII. muscat Functions

Introducción

Aviso
Esta extensión es *EXPERIMENTAL*. Esto significa que el comportamiento de esta extensión, los nombre de sus funciones y en definitiva TODO lo documentado sobre esta extensión, puede cambiar en una futura versión de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Instalación

These functions are only available if PHP was configured with *--with-muscat[=DIR]*.

Tabla de contenidos

muscat_close

(4.0.5 - 4.2.3 only)

muscat_close -- Shuts down the muscat session and releases any memory back to PHP

Description

int **muscat_close** (resource muscat_handle)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

[Not back to the system, note!]

muscat_get

(4.0.5 - 4.2.3 only)

muscat_get -- Gets a line back from the core muscat API

Description

string **muscat_get** (resource muscat_handle)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

Aviso
Esta función puede devolver **`FALSE`**, pero también puede devolver un valor no-booleano que será evaluado **`FALSE`**, como por ejemplo *0* o "". Por favor, lea la sección Booleans para más información. Utilice el operador === para comprobar el valor devuelto por esta función.

muscat_give

(4.0.5 - 4.2.3 only)

muscat_give -- Sends string to the core muscat API

Description

int **muscat_give** (resource muscat_handle, string string)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

muscat_setup_net

(4.0.5 - 4.2.3 only)

muscat_setup_net -- Creates a new muscat session and returns the handle

Description

resource **muscat_setup_net** (string muscat_host)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

muscat_setup_net() creates a new muscat session and returns the handle.

muscat_host is the hostname to connect to. *port* is the port number to connect to.

muscat_setup

(4.0.5 - 4.2.3 only)

muscat_setup -- Creates a new muscat session and returns the handle

Description

resource **muscat_setup** (int size [, string muscat_dir])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.
Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

size is the amount of memory in bytes to allocate for muscat. *muscat_dir* is the muscat installation dir e.g. "/usr/local/empower", it defaults to the compile time muscat directory.

LXXIX. Funciones MySQL

Introducción

Estas funciones le permiten acceder a servidores de bases de datos MySQL. Puede encontrar más información sobre MySQL en http://www.mysql.com/.

La documentación de MySQL puede encontrarse en http://dev.mysql.com/doc/.

Requirimientos

Para contar con éstas funciones, debe compilar PHP con soporte MySQL.

Instalación

Al usar la opción de configuración *--with-mysql[=DIR]*, usted habilita a PHP para que acceda a bases de datos MySQL.

En PHP 4, la opción *--with-mysql* está habilitada por defecto. Para desactivar este comportamiento prestablecido, usted puede usar la opción de configuración *--without-mysql*. También, en PHP 4, si se habilita MySQL sin especificar el directorio de instalación de MySQL, PHP usará las bibliotecas de cliente de MySQL incorporadas. En Windows no existe DLL, simplemente es parte de PHP 4. Los usuarios que ejecutan otras aplicaciones que usan MySQL (auth-mysql, por ejemplo) no deberían usar la biblioteca incorporada, en su lugar deben especificar la ruta al directorio de instalación de MySQL, de este modo: *--with-mysql=/path/to/mysql*. Esto obligará a PHP a usar las bibliotecas de cliente instaladas por MySQL, para así evitar cualquier conflicto.

En PHP 5, el soporte para MySQL no se encuentra habilitado por defecto, ni lo está la biblioteca incorporada con PHP. Lea este FAQ para conocer los detalles del porqué. Debido a esto, los usuarios de Windows deben habilitar `php_mysql.dll` al interior de `php.ini` y copiar `libmysql.dll` en el directorio de sistema de Windows, o ponerlo al alcance de *PATH*. Para la compilación, simplemente use *--with-mysql=[DIR]* en donde *[DIR]* apunta a su directorio de instalación de MySQL.

Esta extensión de MySQL no soporta la funcionalidad completa de versiones de MySQL superiores a 4.1.0. Para ellas, use MySQLi.

Si quisiera instalar la extensión mysql junto con la extensión mysqli, debe usar la misma biblioteca de cliente para evitar cualquier conflicto.

Aviso
Pueden encontrarse problemas de inicialización y bloqueos de PHP cuando esta extensión es cargada en conjunto con la extensión recode. Consulte sobre la extensión recode para más información.

> **Nota:** Si necesita juegos de caracteres diferentes a *latin* (el juego por defecto), tendrá que instalar una biblioteca de mysql externa (no incorporada) que haya sido compilada con soporte para los juegos de caracteres.

Configuración en tiempo de ejecución

El comportamiento de estas funciones está afectado por los valores definidos en `php.ini`.

Tabla 1. Opciones de Configuración MySQL

Nombre	Por defecto	Modificable
mysql.allow_persistent	"On"	PHP_INI_SYSTEM
mysql.max_persistent	"-1"	PHP_INI_SYSTEM
mysql.max_links	"-1"	PHP_INI_SYSTEM
mysql.trace_mode	"Off"	PHP_INI_ALL
mysql.default_port	NULL	PHP_INI_ALL

Nombre	Por defecto	Modificable
mysql.default_socket	NULL	PHP_INI_ALL
mysql.default_host	NULL	PHP_INI_ALL
mysql.default_user	NULL	PHP_INI_ALL
mysql.default_password	NULL	PHP_INI_ALL
mysql.connect_timeout	"0"	PHP_INI_SYSTEM

Para más detalles sobre las constantes PHP_INI_* y sus definiciones, vea **ini_set()**.

A continuación se presenta una corta explicación de las directivas de configuración.

mysql.allow_persistent **boolean**

Indica si se permiten conexiones persistentes con MySQL.

mysql.max_persistent **integer**

El número máximo de conexiones persistentes con MySQL por proceso.

mysql.max_links **integer**

El número máximo de conexiones con MySQL por proceso, incluyendo conexiones persistentes.

mysql.trace_mode **boolean**

Modo de rastreo. Cuando se habilita *mysql.trace_mode*, se muestran advertencias para la apertura de tablas/índices, conjuntos de resultados no liberados, y errores SQL. (Se introdujo en PHP 4.3.0)

mysql.default_port **string**

El número de puerto TCP predeterminado para usar cuando se conecta con el servidor de bases de datos, si no se indica otro. Si no se indica un valor predeterminado, el puerto se obtendrá de la variable de entorno *MYSQL_TCP_PORT*, la entrada *mysql-tcp* en `/etc/services` o la constante de tiempo de compilación **`MYSQL_PORT`**, en ese orden. En Win32 sólo se usa la constante **`MYSQL_PORT`**.

mysql.default_socket **string**

El nombre de socket predeterminado a ser usado cuando se realicen conexiones con un servidor de base de datos local, si no se indica algún otro.

mysql.default_host **string**

La máquina anfitriona predeterminada a ser usada cuando se realicen conexiones con un servidor de bases de datos, si no se indica otro valor. No es aplicable en modo seguro.

mysql.default_user **string**

El nombre de usuario prodeterminado para conectarse al servidor de bases de datos si no se

indica otro. No es aplicable bajo modo seguro.

mysql.default_password **string**

La contraseña predeterminada a usar cuando se realicen conexiones con el servidor de bases de datos, si no se indica otro valor. No es aplicable en modo seguro.

mysql.connect_timeout **integer**

Tiempo de espera máximo de conexión, en segundos. Bajo Linux este tiempo de espera es usado también cuando se espera la primera respuesta del servidor.

Tipos de recursos

Hay dos tipos de recursos usados en el módulo MySQL. El primero es el identificador de enlace para una conexión de base de datos, el segundo es un recurso que almacena el resultado de una consulta.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

A partir de PHP 4.3.0, es posible especificar banderas de cliente adicionales para las funciones **mysql_connect()** y **mysql_pconnect()**. Las siguientes constantes están definidas:

Tabla 2. Constantes de cliente MySQL

Constante	Descripción
MYSQL_CLIENT_COMPRESS	Usar protocolo de compresión
MYSQL_CLIENT_IGNORE_SPACE	Permitir espacios después de nombres de funciones
MYSQL_CLIENT_INTERACTIVE	Permitir tantos segundos de inactividad como indique interactive_timeout (en lugar de wait_timeout) antes de cerrar la conexión
MYSQL_CLIENT_SSL	Usar encripción SSL. Esta bandera se encuentra disponible únicamente con la versión 4.x o más reciente de la biblioteca cliente de MySQL. La versión 3.23.x se distribuye tanto con PHP 4 como con los binarios de Windows de PHP 5.

La función **mysql_fetch_array()** usa una constante para los diferentes tipos de matrices de resultado. Las siguientes constantes están definidas:

Tabla 3. Constantes MySQL-fetch

Constante	Descripción
MYSQL_ASSOC	Las columnas son devueltas en la matriz usando el nombre del campo como índice.
MYSQL_BOTH	Las columnas son devueltas en la matriz teniendo tanto un índice numérico como un índice correspondiente al nombre del campo.
MYSQL_NUM	Las columnas son devueltas en la matriz teniendo un índice numérico a los campos. Este índice comienza en 0, el primer campo del resultado.

Ejemplos

Este sencillo ejemplo muestra cómo conectarse, ejecutar una consulta, imprimir las filas resultantes y desconectarse de una base de datos MySQL.

Ejemplo 1. Ejemplo general de la extensión MySQL

```
<?php
/* Conexion, seleccion de base de datos */
$enlace = mysql_connect("host_mysql", "usuario_mysql", "contrasenya_mysql")
    or die("No pudo conectarse : " . mysql_error());
echo "Conexi&oacute;n exitosa";
mysql_select_db("mi_base_de_datos") or die("No pudo seleccionarse la BD.");

/* Realizar una consulta SQL */
$consulta  = "SELECT * FROM mi_tabla";
$resultado = mysql_query($consulta) or die("La consulta fall&oacute;: " . mysql_error()

/* Impresion de resultados en HTML */
echo "<table>\n";
while ($linea = mysql_fetch_array($resultado, MYSQL_ASSOC)) {
    echo "\t<tr>\n";
    foreach ($linea as $valor_col) {
        echo "\t\t<td>$valor_col</td>\n";
    }
    echo "\t</tr>\n";
}
echo "</table>\n";

/* Liberar conjunto de resultados */
mysql_free_result($resultado);

/* Cerrar la conexion */
mysql_close($enlace);
?>
```

Tabla de contenidos

mysql_fetch_field -- Extrae la información de una columna y la devuelve como un objeto.
mysql_fetch_lengths -- Devuelve la longitud de cada salida en un resultado
mysql_fetch_object -- Extrae una fila de resultado como un objeto
mysql_fetch_row -- Devuelve una fila de resultado como matriz
mysql_field_flags -- Devuelve las banderas asociados con el campo específicado en un resultado
mysql_field_len -- Devuelve la longitud del campo específicado
mysql_field_name -- Devuelve el nombre del campo específicado en un resultado
mysql_field_seek -- Asigna el puntero del resultado al offset del campo específicado
mysql_field_table -- Devuelve el nombre de la tabla donde está el campo específicado
mysql_field_type -- Devuelve el tipo del campo específicado en un resultado
mysql_free_result -- Libera la memoria del resultado
mysql_get_client_info -- Obtener información del cliente MySQL
mysql_get_host_info -- Obtener información de la máquina anfitriona MySQL
mysql_get_proto_info -- Obtener información del protocolo MySQL
mysql_get_server_info -- Obtener información del servidor MySQL
mysql_info -- Obtiene información sobre la consulta más reciente
mysql_insert_id -- Devuelve el identificador generado en la última llamada a INSERT
mysql_list_dbs -- Lista las bases de datos disponibles en el servidor MySQL
mysql_list_fields -- Lista los campos del resultado de MySQL
mysql_list_processes -- Lista los procesos MySQL
mysql_list_tables -- Lista las tablas en una base de datos MySQL
mysql_num_fields -- devuelve el número de campos de un resultado
mysql_num_rows -- Devuelve el número de filas de un resultado
mysql_pconnect -- Abre una conexión persistente al servidor MySQL
mysql_ping -- Efectuar un chequeo de respuesta (ping) sobre una conexión de servidor o reconectarse si no hay conexión
mysql_query -- Envía una consulta de MySQL
mysql_real_escape_string -- Escapa caracteres especiales de una cadena para su uso en una sentencia SQL
mysql_result -- Devuelve datos de un resultado
mysql_select_db -- Selecciona un base de datos MySQL
mysql_stat -- Obtener el status actual del sistema
mysql_tablename -- Devuelve el nombre de la tabla de un campo
mysql_thread_id -- Devuelve el ID del hilo actual
mysql_unbuffered_query -- Envía una consulta SQL a MySQL, sin recuperar ni colocar en búfer las filas de resultado

mysql_affected_rows

(PHP 3, PHP 4 , PHP 5)

mysql_affected_rows -- Devuelve el número de filas afectadas de la última operación MySQL

Descripción

int **mysql_affected_rows** ([int identificador_de_enlace])

mysql_affected_rows() devuelve el número de filas afectadas en la ultima sentencia INSERT, UPDATE o DELETE sobre el servidor asociado con el *identificador_de_enlace* especificado. Si el identificador de enlace no ha sido especificado, se asume por defecto el último enlace.

> **Nota:** Si está usando transacciones, necesitará llamar **mysql_affected_rows()** después del INSERT, UPDATE, o DELETE, no después del commit.

Si la última sentencia fue un DELETE sin clausula WHERE, todos los registros han sido borrados de la tabla pero esta función devolvera cero.

> **Nota:** Cuando se usa UPDATE, MySQL no actualizará las columnas donde el nuevo valor es el mismo al actual. Esto crea la posibilidad de que **mysql_affected_rows()** pueda no ser igual al número de filas encontradas, solo el número de filas que fueron literalmente afectadas por la sentencia.
>
> La sentencia REPLACE primero borra el registro con la misma llave principal y entonces inserta el nuevo registro. Esta función regresa el número de registros borrados más el número de registros insertados.

Para obtener el número de fils regresadas por un SELECT, es posible usar también **mysql_num_rows()**.

Si la última consulta falló, esta función regresará -1.

Ejemplo 1. Delete-Query

```
<?php
/* connect to database */
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
    die('Could not connect: ' . mysql_error());
}
mysql_select_db('mydb');

/* this should return the correct numbers of deleted records */
mysql_query('DELETE FROM mytable WHERE id < 10');
printf("Records deleted: %d\n", mysql_affected_rows());

/* with a where clause that is never true, it should return 0 */
mysql_query('DELETE FROM mytable WHERE 0');
printf("Records deleted: %d\n", mysql_affected_rows());
?>
```

El ejemplo anterior prducirá la siguiente salida:

```
Records deleted: 10
Records deleted: 0
```

Ejemplo 2. Update-Query

```
<?php
/* connect to database */
mysql_connect("localhost", "mysql_user", "mysql_password") or
    die("Could not connect: " . mysql_error());
mysql_select_db("mydb");

/* Update records */
mysql_query("UPDATE mytable SET used=1 WHERE id < 10");
printf ("Updated records: %d\n", mysql_affected_rows());
mysql_query("COMMIT");
?>
```

El ejemplo anterior prducirá la siguiente salida:

```
Updated Records: 10
```

Vea también **mysql_num_rows()**, **mysql_info()**.

mysql_change_user

(PHP 3>= 3.0.13)

mysql_change_user -- Cambia el usuario conectado en la conexión activa

Descripción

int **mysql_change_user** (cadena usuario, cadena password [, cadena base_de_datos [, int identificador_de_enlace]])

mysql_change_user() cambia el usuario conectado en la actual conexión activa, o si se especifica, en la conexión determinada por el *identificador_de_enlace* opcional. Si se especifica la base de datos, ésta será la base por defecto después del cambio de usuario. Si la nueva combinación de usuario/password no está autorizada, el usuario actualmente conectado permanece activo.

> **Nota:** Esta función es obsoleta y solo está disponible en PHP 3 y requiere MySQL 3.23.3 o superior.

mysql_client_encoding

(PHP 4 >= 4.3.0, PHP 5)

mysql_client_encoding -- Devuelve el nombre del juego de caracteres

Descripción

string **mysql_client_encoding** ([resource id_enlace])

mysql_client_encoding() devuelve el nombre del juego de caracteres predeterminado para la conexión actual.

Ejemplo 1. Ejemplo de mysql_client_encoding()

```
<?php
$enlace = mysql_connect('localhost', 'mysql_user', 'mysql_password');

$juego_cars = mysql_client_encoding($enlace);
printf("el juego de caracteres actual es %s\n", $juego_cars);
?>
```

El ejemplo anterior produciría la suguiente salida:

```
el juego de caracteres actual es latin1
```

Vea también **mysql_real_escape_string()**

mysql_close

(PHP 3, PHP 4 , PHP 5)

mysql_close -- cierra el enlace con MySQL

Descripción

int **mysql_close** ([int identificador_de_enlace])

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

mysql_close() cierra el enlace con la base MySQL que esta asociada con el identificador de enlace especificado. Si no se especifica el identificador de enlace, se asume por defecto el último enlace.

> **Nota:** Normalmente no es necesario ya que la aperturas no-persistentes son cerradas automaticamente al final de la ejecución del script. Vea tambié liberar recursos.

mysql_close() no cerrará los enlaces persistentes generados con **mysql_pconnect()**.

Ejemplo 1. Ejemplo de MySQL close

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
    die('Could not connect: ' . mysql_error());
}
echo 'Connected successfully';
mysql_close($link);
?>
```

Ver también: **mysql_connect()**, **mysql_pconnect()**.

mysql_connect

(PHP 3, PHP 4 , PHP 5)

mysql_connect -- Abre una conexión a un servidor MySQL

Descripción

int **mysql_connect** ([cadena hostname [, cadena usuario [, cadena password [, bool new_link [, int client_flags]]]]])

Devuelve: Un identificador de enlace positivo si tiene exito, o falso si error.

mysql_connect()establece una conexión a un servidor MySQL. Todos los argumentos son opcionales, y si no hay, se asumen los valores por defecto ('localhost', usuario propietario del proceso del servidor, password vacío).

El *hostname* puede incluir tambien un número de puerto . ej. "hostname:puerto" o un camino al socket ej. ":/camino/al/socket" para localhost.

> **Nota:** Siempre que específicque "localhost" o "localhost:port" como servidor, la libreria cliente de MySQL evitará esto y tratará de conectarse a socket local (nombrado pipe en Windows). Si quiere usar TCP/IP, use "127.0.0.1" en vez de "localhost". Si las librerias cliente de MySQL intentan conectarse a un socket local ñocado, debe fijar la trayectoria como mysql.default_host en el archivo de configuración PHP y dejar el campo del servidor en blanco.
>
> Soporte para ":puerto" fue añadido en PHP 3.0B4.
>
> Soporte para ":/camino/al/socket" fue añadido en PHP 3.0.10.
>
> Puede suprimir el mensaje de error en caso de falla anteponiendo una @ al nombre de la funciÃ‚Â´n.

Si se hace una segunda llamada a **mysql_connect()** con los mismos argumentos, no se abrirá nuevo enlace, en lugar de eso, se regresa el identificador de enlace ya abierto. El parámetro *new_link* modifica este comportamiento y hace que **mysql_connect()** siempre abrir un nuevo enlace, aún si **mysql_connect()** había sido llamado antes con los mismos parámetros. El pará *client_flags* puede

ser una combinación de las constantes: **MYSQL_CLIENT_COMPRESS**, **MYSQL_CLIENT_IGNORE_SPACE** o **MYSQL_CLIENT_INTERACTIVE**.

> **Nota:** El parámetro *new_link* estuvo disponible en PHP 4.2.0
>
> El parámetro *client_flags* estuvo disponible en PHP 4.3.0

El enlace al servidor sera cerrado tan pronto como la ejecución del script finalice, a menos que se cierre antes explícitamente llamando a **mysql_close()**.

Ejemplo 1. Ejemplo de MySQL connect

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
    die('Could not connect: ' . mysql_error());
}
echo 'Connected successfully';
mysql_close($link);
?>
```

Ver también : **mysql_pconnect()**, **mysql_close()**.

mysql_create_db

(PHP 3, PHP 4 , PHP 5)

mysql_create_db -- Crea una base MySQL

Descripción

int **mysql_create_db** (cadena base_de_datos [, int identificador_de_enlace])

mysql_create_db() intenta crear una base nueva en el servidor asociado al identificador de enlace.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplo 1. Ejemplo de MySQL create

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
    die('Could not connect: ' . mysql_error());
}

if (mysql_create_db('my_db')) {
    echo "Database created successfully\n";
} else {
    echo 'Error creating database: ' . mysql_error() . "\n";
}
?>
```

Por razones de compatibilidad puede usarse **mysql_createdb()** igualmente.

> **Nota:** La función **mysql_create_db()** es obsoleta. Es preferible usar **mysql_query()** para ejecutar una sentencia *SQL CREATE DATABASE* en su lugar.

Aviso

Esta función no será disponible y la extensión de MySQL fue construida/compilada en una libreria de cliente de MySQL 4.X.

Ver también: **mysql_query()**.

mysql_data_seek

(PHP 3, PHP 4 , PHP 5)

mysql_data_seek -- Mueve el puntero interno

Descripción

int **mysql_data_seek** (int id_resultado, int numero_de_fila)

mysql_data_seek() mueve el puntero de fila interno a la fila específicada para el identificador de resultado. La próxima llamada a **mysql_fetch_row()** devolverá esa fila.

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

numero_de_fila empieza en . El *numero_de_fila* debe ser un valor dentro del rango de o a **mysql_num_rows()** - 1. Sin embargo si el resultado está vacío (**mysql_num_rows()** == 0), una búsqueda a 0 fallará con E_WARNING y **mysql_data_seek()** regresará **`FALSE`**.

> **Nota:** La función **mysql_data_seek()** puede ser usada solo en conjunto con **mysql_query()**, no con **mysql_unbuffered_query()**.

Ejemplo 1. Ejemplo de MySQL data seek

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
    die('Could not connect: ' . mysql_error());
}
$db_selected = mysql_select_db('sample_db');
if (!$db_selected) {
    die('Could not select database: ' . mysql_error());
}
$query = 'SELECT last_name, first_name FROM friends';
$result = mysql_query($query);
if (!$result) {
    die('Query failed: ' . mysql_error());
}
/* fetch rows in reverse order */
for ($i = mysql_num_rows($result) - 1; $i >= 0; $i--) {
    if (!mysql_data_seek($result, $i)) {
        echo "Cannot seek to row $i: " . mysql_error() . "\n";
        continue;
    }

    if (!($row = mysql_fetch_assoc($result))) {
        continue;
    }

    echo $row['last_name'] . ' ' . $row['first_name'] . "<br />\n";
}

mysql_free_result($result);
?>
```

Vea también **mysql_query()**, **mysql_num_rows()**, **mysql_fetch_row()**, **mysql_fetch_assoc()**,

mysql_fetch_array(), y mysql_fetch_object().

mysql_db_name

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

mysql_db_name -- Obtener datos de resultado

Descripción

string **mysql_db_name** (resource resultado, int fila [, mixed campo])

mysql_db_name() toma como primer argumento un apuntador al resultado de una llamada a mysql_list_dbs(). El parámetro *fila* es un índice dentro del conjunto de resultados.

Si ocurre algún error, se devuelve **FALSE**. Use mysql_errno() y mysql_error() para determinar la naturaleza del error.

Ejemplo 1. Ejemplo de mysql_db_name()

```
<?php
error_reporting(E_ALL);

$enlace = mysql_connect('dbhost', 'username', 'password');
$lista_db = mysql_list_dbs($enlace);

$i = 0;
$cnt = mysql_num_rows($lista_db);
while ($i < $cnt) {
    echo mysql_db_name($lista_db, $i) . "\n";
    $i++;
}
?>
```

Para efectos de compatibilidad con versiones anteriores, **mysql_dbname()** también puede ser usada. Sin embargo, esta función se ha hecho obsoleta.

Vea también mysql_list_dbs(), y mysql_tablename().

mysql_db_query

(PHP 3, PHP 4 , PHP 5)

mysql_db_query -- Envia una sentencia MySQL al servidor

Descripción

int **mysql_db_query** (cadena base_de_datos, cadena sentencia [, int identificador_de_enlace])

Devuelve: Un identificador de resultado positivo o falso si error. La función también regresa **TRUE**/**FALSE** para las sentencias *INSERT*/*UPDATE*/*DELETE* para indicar exito/falla.

mysql_db_query() selecciona una base y ejecuta una sentencia en ella. Si el identificador de enlace no ha sido especificado, la función intenta encontrar un enlace abierto al servidor MySQL y si no lo encuentra, intetará crear uno como si fuera llamado mysql_connect() sin argumentos

Tenga en cuenta que esta funció **NO** se regresa a la base de datos a la que estaba conectado antes. En otras palabras usted no puede usar esta función para correr *temporalmente* una consulta SQL en otra base de datos, usted debe manualmente regresarse. Los usuarios son animados a usar la sintaxis *database.table* en las consultas SQL en vez de esta función.

Ver también**mysql_connect()**, **mysql_query()**.

> **Nota:** Esta función es obsoleta desde PHP 4.0.6 no use esta función. Use **mysql_select_db()** y **mysql_query()**.

mysql_drop_db

(PHP 3, PHP 4 , PHP 5)

mysql_drop_db -- Borra una base de datos MySQL

Descripción

int **mysql_drop_db** (cadena base_de_datos [, int identificador_de_enlace])

mysql_drop_db() intenta suprimir una base de datos completa del servidor asociado al identificador de enlace.

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Ver también: **mysql_create_db()**. Por razones de compatibilidad puede usarse **mysql_dropdb()** igualmente. Aunque es obsoleta.

> **Nota:** La función **mysql_drop_db()** es obsoleta. Es preferible usar **mysql_query()** para enviar una sentencia SQL *DROP DATABASE* en su lugar.

Aviso
Esta función no está disponible si la extensión de MySQL fue compilada para las librerias cliente de MySQL 4.x.

Vea también **mysql_query()**.

mysql_errno

(PHP 3, PHP 4 , PHP 5)

mysql_errno -- Deuelve el número del mensaje de error de la última operación MySQL

Descripción

int **mysql_errno** ([int identificador_de_enlace])

Regresa el número de error de la última función, o *0* (cero) si no hay error.

Los errores que se obtienen de la base de datos MySQL ya no generan alertas. En lugar de eso, use **mysql_errno()** para obtener el código de error. Note que esta función solo regresa el código de error

de la función MySQLmÃ‚Â´s recientemente executada (sin incluir **mysql_error()** y **mysql_errno ()**), así que si quiere usarla, asegurese de chear el valor antes de llamar a otra funció de MySQL.

Ejemplo 1. Ejemplo de mysql_errno

```
<?php
$link = mysql_connect("localhost", "mysql_user", "mysql_password");

if (!mysql_select_db("nonexistentdb", $link)) {
    echo mysql_errno($link) . ": " . mysql_error($link). "\n";
}

mysql_select_db("kossu", $link);
if (!mysql_query("SELECT * FROM nonexistenttable", $link)) {
    echo mysql_errno($link) . ": " . mysql_error($link) . "\n";
}
?>
```

El ejemplo anterior producirá la siguiente salida:

```
1049: Unknown database 'nonexistentdb'
1146: Table 'kossu.nonexistenttable' doesn't exist
```

> **Nota:** Si el parámetro opcional es específicado el identificador_de_enlace es usado para obtener el código de error. Si no, se usa el último enlace abierto.

Ver también: **mysql_error()** y Códigos de error MySQL.

mysql_error

(PHP 3, PHP 4 , PHP 5)

mysql_error -- Devuelve el texto del mensaje de error de la última operación MySQL

Descripción

cadena **mysql_error** ([int identificador_de_enlace])

Regresa el texto del error de la última funció MySQL o ″(cadena vacía) si no ocurrió error. Si no se específica el identificador de enlace en la función se usa el último enlace abierto exitosamente para obtener el mensaje de error del servidor MySQL.

Los errores que se obtienen de la base de datos MySQL ya no generan alertas. En lugar de eso, use **mysql_errno()** para obtener el código de error. Note que esta función solo regresa el código de error de la función MySQLmÃ‚Â´s recientemente executada (sin incluir **mysql_error()** y **mysql_errno ()**), así que si quiere usarla, asegurese de chear el valor antes de llamar a otra funció de MySQL.

Ejemplo 1. Ejemplo de mysql_error

```
<?php
$link = mysql_connect("localhost", "mysql_user", "mysql_password");

mysql_select_db("nonexistentdb", $link);
echo mysql_errno($link) . ": " . mysql_error($link). "\n";

mysql_select_db("kossu", $link);
mysql_query("SELECT * FROM nonexistenttable", $link);
echo mysql_errno($link) . ": " . mysql_error($link) . "\n";
?>
```

El ejemplo anterior producirá la siguiente salida:

```
1049: Unknown database 'nonexistentdb'
1146: Table 'kossu.nonexistenttable' doesn't exist
```

Ver también: **mysql_errno()** Códigos de error MySQL.

mysql_escape_string

(PHP 4 >= 4.0.3, PHP 5)

mysql_escape_string -- Escapa una cadena para su uso en mysql_query

Descripción

string **mysql_escape_string** (string cadena_no_escapada)

Esta función escapa la *cadena_no_escapada*, de modo que sea seguro usarla en un llamado a **mysql_query()**.

> **Nota: mysql_escape_string()** no escapa % ni _.
>
> Esta función es idéntica a **mysql_real_escape_string()** con la excepción de que **mysql_real_escape_string()** recibe un gestor de conexión y escapa la cadena de acuerdo al juego de caracteres actual. **mysql_escape_string()** no recibe un argumento de conexión, y no hace caso al valor actual del juego de caracteres.

Ejemplo 1. Ejemplo de mysql_escape_string()

```
<?php
$item = "Zak's Laptop";
$item_escapado = mysql_escape_string($item);
printf("La cadena escapada: %s\n", $item_escapado);
?>
```

El anterior ejemplo produciría la siguiente salida:

```
La cadena escapada: Zak\'s Laptop
```

> **Nota:** Esta función se considera obsoleta a partir de PHP 4.3.0. No use esta función. Use **mysql_real_escape_string()** en su lugar.

Vea también **mysql_real_escape_string()**, **addslashes()** y la directiva magic_quotes_gpc.

mysql_fetch_array

(PHP 3, PHP 4 , PHP 5)

mysql_fetch_array -- Extrae la fila de resultado como una matriz asociativa, una matriz numérica o ambas

Descripción

array **mysql_fetch_array** (int id_resultado [, int tipo_de_resultado])

Devuelve una matriz que corresponde a la sentencia extraida, o falso si no quedan más filas.

mysql_fetch_array() es una versión extendida de **mysql_fetch_row()**. Además de guardar los datos en el índice numérico de la matriz, guarda también los datos en los índices asociativos, usando el nombre de campo como clave.

Si dos o más columnas del resultado tienen el mismo nombre de campo, la última columna toma la

prioridad. Para acceder a la(s) otra(s) columna(s) con el mismo nombre, se debe escificar el indice numerico o definir un alias para la columna. En columnas con alias, usted no puede acceder al contenido con el nombre original de la columna (usando *'field'* en este ejemplo)

Ejemplo 1. Consulta con campos repetidos usando alias

```
SELECT table1.field AS foo, table2.field AS bar FROM table1, table2
```

La función **mysql_fetch_array()** no es significativemente mas lenta que **mysql_fetch_row()**, sin embargo tiene un valor añadido importante.

El segundo argumento opcional *tipo_de_resultado* en **mysql_fetch_array()** es una constante y puede tomar los siguientes valores: **MYSQL_ASSOC**, **MYSQL_NUM**, y **MYSQL_BOTH**. Esta caraterística fue agregada en PHP 3.0.7. **MYSQL_BOTH**

Usando **MYSQL_BOTH**, usted obtendrá una matrix con índices asociativos y numéricos. Usando **MYSQL_ASSOC**, usted solo tendrá índices asociativos (tal como funciona **mysql_fetch_assoc()**), usando **MYSQL_NUM**, solo obtendrá los índices numéricos (tal como si fuera **mysql_fetch_row()**).

> **Nota:** Los nombres de campos retornados por esta función diferencian entre *mayusculas y minusculas*.
>
> **Nota:** Esta funcion define campos NULL como valores PHP **NULL**.

Vea también **mysql_fetch_row()**, **mysql_fetch_assoc()**, **mysql_data_seek()**, y **mysql_query()**.

Ejemplo 2. mysql_fetch_array() con MYSQL_NUM

```
<?php
mysql_connect("localhost", "mysql_user", "mysql_password") or
    die("Could not connect: " . mysql_error());
mysql_select_db("mydb");

$result = mysql_query("SELECT id, name FROM mytable");

while ($row = mysql_fetch_array($result, MYSQL_NUM)) {
    printf("ID: %s  Name: %s", $row[0], $row[1]);
}

mysql_free_result($result);
?>
```

Ejemplo 3. mysql_fetch_array() con MYSQL_ASSOC

```
<?php
mysql_connect("localhost", "mysql_user", "mysql_password") or
    die("Could not connect: " . mysql_error());
mysql_select_db("mydb");

$result = mysql_query("SELECT id, name FROM mytable");

while ($row = mysql_fetch_array($result, MYSQL_ASSOC)) {
    printf("ID: %s  Name: %s", $row["id"], $row["name"]);
}

mysql_free_result($result);
?>
```

Ejemplo 4. mysql_fetch_array() con MYSQL_BOTH

```
<?php
mysql_connect("localhost", "mysql_user", "mysql_password") or
    die("Could not connect: " . mysql_error());
mysql_select_db("mydb");

$result = mysql_query("SELECT id, name FROM mytable");

while ($row = mysql_fetch_array($result, MYSQL_BOTH)) {
    printf ("ID: %s  Name: %s", $row[0], $row["name"]);
}

mysql_free_result($result);
?>
```

mysql_fetch_assoc

(PHP 4 >= 4.0.3, PHP 5)

mysql_fetch_assoc -- Recupera una fila de resultado como una matriz asociativa

Descripción

array **mysql_fetch_assoc** (resource resultado)

Devuelve una matriz asociativa que corresponde a la fila recuperada, o **FALSE** si no hay más filas.

mysql_fetch_assoc() es ñalente a llamar **mysql_fetch_array()** con MYSQL_ASSOC como segundo parámetro opcional. Ã*f*Å¡nicamente devuelve una matriz asociativa. Esta es la forma en que originalmente trabajaba **mysql_fetch_array()**. Si necesita índices numéricos así como asociativos, utilice **mysql_fetch_array()**.

Si dos o más columnas del resultado tienen los mismos nombres de campo, la última columna tomará precedencia. Para acceder a otras columnas con el mismo nombre, tendrá que acceder al resultado con índices numéricos mediante el uso de **mysql_fetch_row()** o agregar sobrenombres. Vea el ejemplo en la descripción de **mysql_fetch_array()** respecto a los sobrenombres.

Algo importante a notar es que el uso de **mysql_fetch_assoc()** *no es significativamente* más lento que el uso de **mysql_fetch_row()**, al mismo tiempo que provee un valor agregado considerable.

> **Nota:** Los nombres de campos retornados por esta función diferencian entre *mayusculas y minusculas*.
>
> **Nota:** Esta funcion define campos NULL como valores PHP **NULL**.

Ejemplo 1. Un ejemplo extendido de mysql_fetch_assoc()

```php
<?php

$conexion = mysql_connect("localhost", "mysql_user", "mysql_password");

if (!$conexion) {
    echo "No pudo conectarse a la BD: " . mysql_error();
    exit;
}

if (!mysql_select_db("nombre_de_la_bd")) {
    echo "No ha sido posible seleccionar la BD: " . mysql_error();
    exit;
}

$sql = "SELECT id as id_usuario, nombre_completo, status_usuario
        FROM   alguna_tabla
        WHERE  status_usuario = 1";

$resultado = mysql_query($sql);

if (!$resultado) {
    echo "No pudo ejecutarse satisfactoriamente la consulta ($sql) " .
         "en la BD: " . mysql_error();
    exit;
}

if (mysql_num_rows($resultado) == 0) {
    echo "No se han encontrado filas, nada a imprimir, asi que voy " .
         "a detenerme.";
    exit;
}

// Mientras exista una fila de datos, colocar esa fila en $fila
// como una matriz asociativa
// Nota: Si solo espera una fila, no hay necesidad de usar un ciclo
// Nota: Si coloca extract($fila); dentro del siguiente ciclo,
//       estara creando $id_usuario, $nombre_completo, y $status_usuario
while ($fila = mysql_fetch_assoc($resultado)) {
    echo $fila["id_usuario"];
    echo $fila["nombre_completo"];
    echo $fila["status_usuario"];
}

mysql_free_result($resultado);

?>
```

Vea también **mysql_fetch_row()**, **mysql_fetch_array()**, **mysql_query()** y **mysql_error()**.

mysql_fetch_field

(PHP 3, PHP 4 , PHP 5)

mysql_fetch_field -- Extrae la información de una columna y la devuelve como un objeto.

Descripción

objeto **mysql_fetch_field** (int id_resultado [, int salto])

Devuelve un objeto que contiene la información del campo.

Puede usarse **mysql_fetch_field()** para obtener información sobre campos en un resultado. Si no se específica el salto, se extrae el siguiente campo que todavía no ha sido extraido. con **mysql_fetch_field()**.

Las propiedades del objeto son:

- name - nombre de la columna
- table - name de la tabla a la que pertenece la columna
- max_length - longitud máxima de la columna
- not_null - 1 si la columna no puede contener un valor nulo
- primary_key - 1 si la columna es clave primaria
- unique_key - 1 si la columna es clave única
- multiple_key - 1 si la columna es clave no única
- numeric - 1 si la columna es numérica
- blob - 1 si la columna es un BLOB
- type - el tipo de la columna
- unsigned - 1 si la columna es unsigned
- zerofill - 1 si la columna es zero-filled

Ver también **mysql_field_seek()**

> **Nota:** Los nombres de campos retornados por esta función diferencian entre *mayusculas y minusculas*.

Ejemplo 1. Ejemplo mysql_fetch_field

```
<?php
$conn = mysql_connect('localhost:3306', 'user', 'password');
if (!$conn) {
    die('Could not connect: ' . mysql_error());
}
mysql_select_db('database');
$result = mysql_query('select * from table');
if (!$result) {
    die('Query failed: ' . mysql_error());
}
/* get column metadata */
$i = 0;
while ($i < mysql_num_fields($result)) {
    echo "Information for column $i:<br />\n";
    $meta = mysql_fetch_field($result, $i);
    if (!$meta) {
        echo "No information available<br />\n";
    }
    echo "<pre>
blob:         $meta->blob
max_length:   $meta->max_length
multiple_key: $meta->multiple_key
name:         $meta->name
not_null:     $meta->not_null
numeric:      $meta->numeric
primary_key:  $meta->primary_key
table:        $meta->table
type:         $meta->type
unique_key:   $meta->unique_key
unsigned:     $meta->unsigned
zerofill:     $meta->zerofill
</pre>";
    $i++;
}
mysql_free_result($result);
?>
```

mysql_fetch_lengths

(PHP 3, PHP 4 , PHP 5)

mysql_fetch_lengths -- Devuelve la longitud de cada salida en un resultado

Descripción

array **mysql_fetch_lengths** (int id_resultado)

Devuelve: Una matriz que contiene las longitudes de cada campo de la última fila extraida por **mysql_fetch_row()**, o **FALSE** si error.

mysql_fetch_lengths() almacena las longitudes de cada columna en la última fila devuelta por **mysql_fetch_row()**, **mysql_fetch_assoc()**, **mysql_fetch_array()**, y **mysql_fetch_object()** en una matriz, empezando por 0.

Vea también **mysql_field_len()**, **mysql_fetch_row()**, **strlen()**.

Ejemplo 1. Ejemplo de mysql_fetch_lengths

```
<?php
$result = mysql_query("SELECT id,email FROM people WHERE id = '42'");
if (!$result) {
    echo 'Could not run query: ' . mysql_error();
    exit;
}
$row     = mysql_fetch_assoc($result);
$lengths = mysql_fetch_lengths($result);

print_r($row);
print_r($lengths);

?>
```

El resultado del ejemplo seria algo similar a:

```
Array
(
    [id] => 42
    [email] => user@example.com
)
Array
(
    [0] => 2
    [1] => 16
)
```

mysql_fetch_object

(PHP 3, PHP 4 , PHP 5)

mysql_fetch_object -- Extrae una fila de resultado como un objeto

Descripción

objeto **mysql_fetch_object** (int id_resultado)

Devuelve un objeto con las propiedades que corresponden a la última fila extraida, o **FALSE** si no quedan más filas.

mysql_fetch_object() es similar a **mysql_fetch_array()**, con la diferencia que un objeto es devuelto en lugar de una matriz. Indirectamente, quiere decir que solo se puede aceder a los datos por el nombre del campo, y no por su posición. (los números no pueden ser nombres de propiedades de objetos).

> **Nota:** Los nombres de campos retornados por esta función diferencian entre *mayusculas y minusculas*.

> **Nota:** Esta funcion define campos NULL como valores PHP **NULL**.

Ejemplo 1. Ejemplo demysql_fetch_object()

```
<?php

$row = mysql_fetch_object($result);

/* this is valid */
echo $row->field;
/* this is invalid */
//echo $row->0;

?>
```

La función es identica a **mysql_fetch_array()**, y casi tan rápida como **mysql_fetch_row()** (la diferencia es insignificante).

Ejemplo 2. Ejemplo demysql_fetch_object()

```
<?php
mysql_connect("hostname", "user", "password");
mysql_select_db("mydb");
$result = mysql_query("select * from mytable");
while ($row = mysql_fetch_object($result)) {
    echo $row->user_id;
    echo $row->fullname;
}
mysql_free_result($result);
?>
```

Ver también: **mysql_fetch_array()**, **mysql_fetch_assoc()**, **mysql_fetch_row()**, **mysql_data_seek()** y **mysql_query()**.

mysql_fetch_row

(PHP 3, PHP 4 , PHP 5)

mysql_fetch_row -- Devuelve una fila de resultado como matriz

Descripción

array **mysql_fetch_row** (int id_resultado)

Devuelve: Una matriz que corresponde a la fila selecionada, o **FALSE** si no quedan más filas.

> **Nota:** Esta funcion define campos NULL como valores PHP **NULL**.

mysql_fetch_row() seleciona una fila de datos del resultado asociado al identificador de resultado especificado. La fila es devuelta como una matriz. Cada columna del resultdo es guardada en un offset de la matriz, empezando por el offset 0.

La llamada a **mysql_fetch_row()** debería devolver la próxima fila del resultado, o **FALSE** si no quedan más filas.

Ejemplo 1. Ejemplo de mysql_fetch_row

```
<?php
$result = mysql_query("SELECT id,email FROM people WHERE id = '42'");
if (!$result) {
    echo 'Could not run query: ' . mysql_error();
    exit;
}
$row = mysql_fetch_row($result);

echo $row[0]; // 42
echo $row[1]; // the email value
?>
```

Ver también: **mysql_fetch_array()**, **mysql_fetch_object()**, **mysql_data_seek()**, **mysql_fetch_lengths()**, **mysql_result()**.

mysql_field_flags

(PHP 3, PHP 4 , PHP 5)

mysql_field_flags -- Devuelve las banderas asociados con el campo específicado en un resultado

Descripción

cadena **mysql_field_flags** (int id_resultado, int offset_del_campo)

mysql_field_flags() devuelve las banderas del campo específicado. Cada bandera es devuelta como una palabra y están separados por un único espacio, se puede dividir el resultado devuelto utilizando **explode()**.

Soporta las siguientes banderas: *"not_null"*, *"primary_key"*, *"unique_key"*, *"multiple_key"*, *"blob"*, *"unsigned"*, *"zerofill"*, *"binary"*, *"enum"*, *"auto_increment"* y *"timestamp"*.

Ejemplo 1. Ejemplo de mysql_field_flags

```
<?php
$result = mysql_query("SELECT id,email FROM people WHERE id = '42'");
if (!$result) {
    echo 'Could not run query: ' . mysql_error();
    exit;
}
$flags = mysql_field_flags($result, 0);

echo $flags;
print_r(explode(' ', $flags));
?>
```

El resultado del ejemplo seria algo similar a:

```
not_null primary_key auto_increment
Array
(
    [0] => not_null
    [1] => primary_key
    [2] => auto_increment
)
```

Por razones de compatibilidad puede usarse también **mysql_fieldflags()**. Sin embargo esta funció es obsoleta.

mysql_field_len

(PHP 3, PHP 4 , PHP 5)

mysql_field_len -- Devuelve la longitud del campo específicado

Descripción

int **mysql_field_len** (int id_resultado, int offset_del_campo)

mysql_field_len() devuelve la longitud del campo específicado.

Ejemplo 1. Ejemplo de mysql_fetch_len

```php
<?php
$result = mysql_query("SELECT id,email FROM people WHERE id = '42'");
if (!$result) {
    echo 'Could not run query: ' . mysql_error();
    exit;
}

// Will get the length of the value in email so for example
// user@example.com would give us a length of 16
$length = mysql_field_len($result, 'email');
echo $length;
?>
```

Por razones de compatibilidad puede usarse también **mysql_fieldlen()**. Aunque esta función es obsoleta.

Vea tambié **mysql_fetch_lengths()**, **strlen()**.

mysql_field_name

(PHP 3, PHP 4 , PHP 5)

mysql_field_name -- Devuelve el nombre del campo específicado en un resultado

Descripción

cadena **mysql_field_name** (int id_resultado, int indice_del_campo)

mysql_field_name() devuelve el nombre del campo específicado. Los argumentos de la función son el identificador de resultado y el índice del campo.

> **Nota:** *indice_del_campo* comienza en 0.
>
> Por ejemplo el índice del tercer campo será en realidad 2, el índice del cuarto campo será 3 y asi sucesivamente.
>
> **Nota:** Los nombres de campos retornados por esta función diferencian entre *mayusculas y minusculas*.

Ejemplo 1. Ejemplo mysql_field_name

```php
<?php
/* The users table consists of three fields:
 *   user_id
 *   username
 *   password.
 */
$link = @mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
    die('Could not connect to MySQL server: ' . mysql_error());
}
$dbname = 'mydb';
$db_selected = mysql_select_db($dbname, $link);
if (!$db_selected) {
    die('Could not set $dbname: ' . mysql_error());
}
$res = mysql_query('select * from users', $link);

echo mysql_field_name($res, 0) . "\n";
echo mysql_field_name($res, 2);
?>
```

El ejemplo anterior producirá la siguiente salida:

```
user_id
password
```

Por razones de compatibilidad puede usarse también **mysql_fieldname()**. Sin embargo esta función es obsoleta.

mysql_field_seek

(PHP 3, PHP 4 , PHP 5)

mysql_field_seek -- Asigna el puntero del resultado al offset del campo específicado

Descripción

int **mysql_field_seek** (int id_resultado, int offset_del_campo)

Busca el offset del campo específicado. Si la próxima llamada a **mysql_fetch_field()** no incluye un offset de campo, se devolverá ese campo.

Ver también **mysql_fetch_field()**.

mysql_field_table

(PHP 3, PHP 4 , PHP 5)

mysql_field_table -- Devuelve el nombre de la tabla donde está el campo específicado

Descripción

cadena **mysql_field_table** (int id_resultado, int offset_del_campo)

Devuelve el nombre de la tabla donde está el campo.

Ejemplo 1. Ejemplo mysql_field_table

```
<?php
$result = mysql_query("SELECT name,comment FROM people,comments");
if (!$result) {
    echo 'Could not run query: ' . mysql_error();
    exit;
}

// Assuming name is in the people table
$table = mysql_field_table($result, 'name');
echo $table; // people
?>
```

Por razones de compatibilidad puede usarse también **mysql_fieldtable()**. Sin embargo esta función es obsoleta.

Vea también **mysql_list_tables()**.

mysql_field_type

(PHP 3, PHP 4 , PHP 5)

mysql_field_type -- Devuelve el tipo del campo específicado en un resultado

Descripción

cadena **mysql_field_type** (int id_resultado, int offset_del_campo)

mysql_field_type() es similar a la función **mysql_field_name()**. Los argumentos son idénticos, pero se devuelve el tipo de campo. El tipo será *"int"*, *"real"*, *"string"*, *"blob"*, u otros detallados en la documentación de MySQL.

Ejemplo 1. mysql_field_type() example

```
<?php
mysql_connect("localhost", "mysql_username", "mysql_password");
mysql_select_db("mysql");
$result = mysql_query("SELECT * FROM func");
$fields = mysql_num_fields($result);
$rows   = mysql_num_rows($result);
$table = mysql_field_table($result, 0);
echo "Your '" . $table . "' table has " . $fields . " fields and " . $rows . " record(s
echo "The table has the following fields:\n";
for ($i=0; $i < $fields; $i++) {
    $type  = mysql_field_type($result, $i);
    $name  = mysql_field_name($result, $i);
    $len   = mysql_field_len($result, $i);
    $flags = mysql_field_flags($result, $i);
    echo $type . " " . $name . " " . $len . " " . $flags . "\n";
}
mysql_free_result($result);
mysql_close();
?>
```

El ejemplo anterior producirá la siguiente salida:

```
Your 'func' table has 4 fields and 1 record(s)
The table has the following fields:
string name 64 not_null primary_key binary
int ret 1 not_null
string dl 128 not_null
string type 9 not_null enum
```

Por razones de compatibilidad puede usarse también **mysql_fieldtype()**. Aunque esta función es obsoleta.

mysql_free_result

(PHP 3, PHP 4 , PHP 5)

mysql_free_result -- Libera la memoria del resultado

Descripción

int **mysql_free_result** (int id_resultado)

mysql_free_result() solo necesita ser llamada si te preocupa usar demasiado memoria durante la ejecución de tu script. Toda la memoria usada por resultado específicado en el parámetro del identificador de resultado será automáticamente liberada.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

If a non-resource is used for the *result*, an error of level E_WARNING will be emitted. It's worth noting that **mysql_query()** only returns a **resource** for SELECT, SHOW, EXPLAIN, and DESCRIBE queries.

Ejemplo 1. Ejemplo de mysql_free_result

```
<?php
$result = mysql_query("SELECT id,email FROM people WHERE id = '42'");
if (!$result) {
    echo 'Could not run query: ' . mysql_error();
    exit;
}
/* Use the result, assuming we're done with it afterwords */
$row = mysql_fetch_assoc($result);

/* Now we free up the result and continue on with our script */
mysql_free_result($result);

echo $row['id'];
echo $row['email'];
?>
```

Por razones de compatibilidad puede usarse tambien **mysql_freeresult()**. Sin embargo esta función es obsoleta.

Vea también **mysql_query()**, **is_resource()**.

mysql_get_client_info

(PHP 4 >= 4.0.5, PHP 5)

mysql_get_client_info -- Obtener información del cliente MySQL

Descripción

string **mysql_get_client_info** (void)

mysql_get_client_info() devuelve una cadena que representa la versión de la biblioteca cliente.

Ejemplo 1. Ejemplo de mysql_get_client_info()

```
<?php
printf("Informaci&oacute;n del cliente MySQL: %s\n", mysql_get_client_info());
?>
```

El anterior ejemplo produciría la siguiente salida:

```
Informaci&oacute;n del cliente MySQL: 3.23.39
```

Vea también **mysql_get_host_info()**, **mysql_get_proto_info()** y **mysql_get_server_info()**.

mysql_get_host_info

(PHP 4 >= 4.0.5, PHP 5)

mysql_get_host_info -- Obtener información de la máquina anfitriona MySQL

Descripción

string **mysql_get_host_info** ([resource id_enlace])

mysql_get_host_info() devuelve una cadena que describe el tipo de conexión en uso con *id_enlace*, incluyendo el nombre del equipo servidor anfitrión. Si se omite *id_enlace*, se usará la última conexión abierta.

Ejemplo 1. Ejemplo de mysql_get_host_info()

```
<?php
$enlace = mysql_connect("localhost", "mysql_user", "mysql_password");
if (!$enlace) {
    die("No pudo conectarse: " . mysql_error());
}
printf("Informaci&oacute;n del host MySQL: %s\n", mysql_get_host_info());
?>
```

El ejemplo anterior produciría la siguiente salida:

```
Informaci&oacute;n del equipo anfitri&oacute;n MySQL: Localhost via UNIX socket
```

Vea también **mysql_get_client_info()**, **mysql_get_proto_info()** y **mysql_get_server_info()**.

mysql_get_proto_info

(PHP 4 >= 4.0.5, PHP 5)

mysql_get_proto_info -- Obtener información del protocolo MySQL

Descripción

int **mysql_get_proto_info** ([resource id_enlace])

mysql_get_proto_info() devuelve la versión del protocolo usada por la conexión *id_enlace*. Si se omite *id_enlace*, se usará la última conexión abierta.

Ejemplo 1. Ejemplo de mysql_get_proto_info()

```
<?php
$enlace = mysql_connect("localhost", "mysql_user", "mysql_password");
if (!$enlace) {
    die("No pudo conectarse: " . mysql_error());
}
printf("Versi&oacute;n del protocolo MySQL: %s\n", mysql_get_proto_info());
?>
```

El ejemplo anterior produciría la siguiente salida:

```
Versi&oacute;n del protocolo MySQL: 10
```

Vea también **mysql_get_client_info()**, **mysql_get_host_info()** y **mysql_get_server_info()**.

mysql_get_server_info

(PHP 4 >= 4.0.5, PHP 5)

mysql_get_server_info -- Obtener información del servidor MySQL

Descripción

string **mysql_get_server_info** ([resource id_enlace])

mysql_get_server_info() devuelve la versión del servidor usado en la conexión *id_enlace*. Si se omite *id_enlace*, será usada la última conexión abierta.

Ejemplo 1. Ejemplo de mysql_get_server_info()

```
<?php
$enlace = mysql_connect("localhost", "mysql_user", "mysql_password");
if (!$enlace) {
    die("No pudo conectarse: " . mysql_error());
}
printf("Versi&oacute;n del servidor MySQL: %s\n", mysql_get_server_info());
?>
```

El anterior ejemplo produciría la siguiente salida:

```
Versi&oacute;n del servidor MySQL: 4.0.1-alpha
```

Vea también **mysql_get_client_info()**, **mysql_get_host_info()**, **mysql_get_proto_info()**, y **phpversion()**.

mysql_info

(PHP 4 >= 4.3.0, PHP 5)

mysql_info -- Obtiene información sobre la consulta más reciente

Descripción

string **mysql_info** ([resource id_enlace])

mysql_info() devuelve información detallada sobre la última consulta realizada usando el *id_enlace* dado. Si no se indica un *id_enlace*, se asumirá el último enlace abierto.

mysql_info() devuelve una cadena para todas las sentencias listadas a continuación. En cualquier otro caso, devuelve **FALSE**. El formato de la cadena depende en la sentencia dada.

Ejemplo 1. Sentencias MySQL relevantes

```
INSERT INTO ... SELECT ...
String format: Records: 23 Duplicates: 0 Warnings: 0
INSERT INTO ... VALUES (...),(...),(...)...
String format: Records: 37 Duplicates: 0 Warnings: 0
LOAD DATA INFILE ...
String format: Records: 42 Deleted: 0 Skipped: 0 Warnings: 0
ALTER TABLE
String format: Records: 60 Duplicates: 0 Warnings: 0
UPDATE
String format: Rows matched: 65 Changed: 65 Warnings: 0
```

Los números presentados son sólo de caracter ilustrativo; sus valores corresponderán a la consulta usada.

> **Nota: mysql_info()** devuelve un valor diferente a **FALSE** para la sentencia INSERT ... VALUES sólo si se indican múltiples listas de valores en la sentencia.

Vea también **mysql_affected_rows()**, **mysql_insert_id()**, y **mysql_stat()**.

mysql_insert_id

(PHP 3, PHP 4 , PHP 5)

mysql_insert_id -- Devuelve el identificador generado en la última llamada a INSERT

Descripción

int **mysql_insert_id** ([int identificador_de_enlace])

mysql_insert_id() devuelve el identificador generado para un campo de tipo AUTO_INCREMENTED. Se devolverá el identificador generado por el último INSERT para el *identificador_de_enlace*. Si no se específica el *identificador_de_enlace*, se asume por defecto el último enlace abierto.

mysql_insert_id() regresa 0 si la consulta previa no generó un valor AUTO_INCREMENT. Si necesita guardar el valor para un uso posterior, asegurese de llamar **mysql_insert_id()** inmediatamente después de la consulta que generó el valor.

> **Nota:** El valor de la función SQL de MYSQL *LAST_INSERT_ID()* siempre contiene el valor AUTO_INCREMENT má recientemente generado, y no se pierde su valor entre consultas.

Aviso
mysql_insert_id() convierte el tipo de la función nativa de MySQL en el API de C *mysql_insert_id()* a un tipo de *long* (llamada **int** en PHP). Si tu columna AUTO_INCREMENT tiene un tipo BIGINT, el valor regresado por **mysql_insert_id()** será incorrecto. En ese caso, usa la función interna *LAST_INSERT_ID()* de SQL en MySQL en una consulta SQL.

Ejemplo 1. Ejemplo mysql_insert_id

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
    die('Could not connect: ' . mysql_error());
}
mysql_select_db('mydb');

mysql_query("INSERT INTO mytable (product) values ('kossu')");
printf("Last inserted record has id %d\n", mysql_insert_id());
?>
```

Vea también **mysql_query()**, **mysql_info()**.

mysql_list_dbs

(PHP 3, PHP 4 , PHP 5)

mysql_list_dbs -- Lista las bases de datos disponibles en el servidor MySQL

Descripción

int **mysql_list_dbs** ([int identificador_de_enlace])

mysql_list_dbs() devuelve un puntero de resultado que contiene las bases disponibles en el actual

demonio mysql. Utiliza la función **mysql_tablename()** para explotar el puntero de resultado.

Ejemplo 1. Ejemplo mysql_list_dbs

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
$db_list = mysql_list_dbs($link);

while ($row = mysql_fetch_object($db_list)) {
    echo $row->Database . "\n";
}
?>
```

El ejemplo anterior producirá la siguiente salida:

```
database1
database2
database3
...
```

> **Nota:** El código anterior puede funcionar igualmente con **mysql_fetch_row()** u otra funció similar.

Por razones de compatibilidad puede usarse también **mysql_listdbs()**. Sin embargo esta función es obsoleta.

Vea también **mysql_db_name()**, **mysql_select_db()**.

mysql_list_fields

(PHP 3, PHP 4 , PHP 5)

mysql_list_fields -- Lista los campos del resultado de MySQL

Descripción

int **mysql_list_fields** (cadena base_de_datos, cadena tabla [, int dentificador_de_enlace])

> **Nota:** La función **mysql_list_fields()** es onsoleta. Es preferible usar **mysql_query()** para ejecutar una sentencia SQL *SHOW COLUMNS FROM table [LIKE 'name']*.

mysql_list_fields() lista información sobre la tabla. Los argumentos son la base de datos y el nombre de la tabla. Se devuelve un puntero que puede ser usado por las funciones **mysql_field_flags()**, **mysql_field_len()**, **mysql_field_name()**, y **mysql_field_type()**.

Un identificador de resultado es un entero positivo. La función devuelve -1 si se produce un error. Una cadena de caracteres describiendo el error será introducida en *$phperrmsg*, y a menos que la función sea llamada como *@mysql()* el literal del error también será desplegado.

Ejemplo 1. Ejemplo alternativo a mysql_list_fields

```
<?php
 $result = mysql_query("SHOW COLUMNS FROM sometable");
 if (!$result) {
     echo 'Could not run query: ' . mysql_error();
     exit;
 }
 if (mysql_num_rows($result) > 0) {
     while ($row = mysql_fetch_assoc($result)) {
         print_r($row);
     }

?>
```

El ejemplo anterior producirá una salida similar a:

```
Array
 (
     [Field] => id
     [Type] => int(7)
     [Null] =>
     [Key] => PRI
     [Default] =>
     [Extra] => auto_increment
 )
 Array
 (
     [Field] => email
     [Type] => varchar(100)
     [Null] =>
     [Key] =>
     [Default] =>
     [Extra] =>
 )
```

Por razones de compatibilidad puede usarse también **mysql_listfields()**. Sin embargo esta funció es obsoleta.

Vea también **mysql_field_flags()**, **mysql_info()**.

mysql_list_processes

(PHP 4 >= 4.3.0, PHP 5)

mysql_list_processes -- Lista los procesos MySQL

Descripción

resource **mysql_list_processes** ([resource id_enlace])

mysql_list_processes() devuelve un apuntador a un resultado que describe los hilos actuales del servidor.

Ejemplo 1. Ejemplo de mysql_list_processes()

```
<?php
$enlace = mysql_connect('localhost', 'mysql_user', 'mysql_password');

$resultado = mysql_list_processes($enlace);

while ($fila = mysql_fetch_assoc($resultado)){
    printf("%s %s %s %s %s\n", $fila["Id"], $fila["Host"], $fila["db"],
       $fila["Command"], $fila["Time"]);
}
mysql_free_result($resultado);
?>
```

El ejemplo anterior produciría la siguiente salida:

```
1 localhost test Processlist 0
4 localhost mysql sleep 5
```

Vea también **mysql_thread_id()** y **mysql_stat()**.

mysql_list_tables

(PHP 3, PHP 4 , PHP 5)

mysql_list_tables -- Lista las tablas en una base de datos MySQL

Descripción

int **mysql_list_tables** (cadena base_de_datos [, int identificador_de_enlace])

mysql_list_tables() toma el nombre de la base y devuelve un puntero de resultado como la función **mysql_db_query()**. La función **mysql_tablename()** debe ser usada para extraer los nombres de las tablas del puntero.

El parámetro *base_de_datos* es el nombre de la base de datos de la cual se debe extraer la lista de la tabla, a menos que ocurra un error **mysql_list_tables()** regresa **FALSE**

Por razones de compatibilidad puede usarse también **mysql_listtables()**. Esta función es obsoleta y no se recomienda.

> **Nota:** La función **mysql_list_tables()** es obsoleta. Es preferible usar **mysql_query()** para enviar una sentencia SQL *SHOW TABLES [FROM db_name] [LIKE 'pattern']*.

Ejemplo 1. Ejemplo mysql_list_tables

```
<?php
$dbname = 'mysql_dbname';

if (!mysql_connect('mysql_host', 'mysql_user', 'mysql_password')) {
    echo 'Could not connect to mysql';
    exit;
}

$result = mysql_list_tables($dbname);

if (!$result) {
    echo "DB Error, could not list tables\n";
    echo 'MySQL Error: ' . mysql_error();
    exit;
}

while ($row = mysql_fetch_row($result)) {
    echo "Table: $row[0]\n";
}

mysql_free_result($result);
?>
```

Vea también: **mysql_list_dbs()**, **mysql_tablename()**.

mysql_num_fields

(PHP 3, PHP 4 , PHP 5)

mysql_num_fields -- devuelve el número de campos de un resultado

Descripción

int **mysql_num_fields** (int id_resultado)

mysql_num_fields() devuelve el número de campos de un identificador de resultado.

Ver también: **mysql_db_query()**, **mysql_query()**, **mysql_fetch_field()**, **mysql_num_rows()**.

Ejemplo 1. Ejemplo de mysql_num_fields()

```
<?php
$result = mysql_query("SELECT id,email FROM people WHERE id = '42'");
if (!$result) {
    echo 'Could not run query: ' . mysql_error();
    exit;
}
/* returns 2 because id,email === two fields */
echo mysql_num_fields($result);
?>
```

Por razones de compatibilidad puede usarse también **mysql_numfields()**. Sin embargo esta función es obsoleta.

mysql_num_rows

(PHP 3, PHP 4 , PHP 5)

mysql_num_rows -- Devuelve el número de filas de un resultado

Descripción

int **mysql_num_rows** (int id_resultado)

mysql_num_rows() regresa el número de filas en un resultado. Este comando es valido solo para las sentencias SELECT. Para obtener el número de filas afectadas por las sentencias INSERT, UPDATE o DELETE, use **mysql_affected_rows()**.

Ejemplo 1. Ejemplo mysql_num_rows

```
<?php

$link = mysql_connect("localhost", "mysql_user", "mysql_password");
mysql_select_db("database", $link);

$result = mysql_query("SELECT * FROM table1", $link);
$num_rows = mysql_num_rows($result);

echo "$num_rows Rows\n";

?>
```

> **Nota:** Si usa **mysql_unbuffered_query()**, **mysql_num_rows()** no regresará el valor correcto hasta que todas las filas en el resultado hayan sido recuperadas.

Ver también: **mysql_affected_rows()**, **mysql_connect()**, **mysql_data_seek()**, **mysql_select_db()**, **mysql_query()**.

Por razones de compatibilidad puede usarse también **mysql_numrows()**. Sin embargo esta fucnión

mysql_pconnect

(PHP 3, PHP 4 , PHP 5)

mysql_pconnect -- Abre una conexión persistente al servidor MySQL

Descripción

int **mysql_pconnect** ([cadena hostname [, cadena usuario [, cadena password]]])

Devuelve: un identificador de enlace persistente, o **`FALSE`** si se produce un error.

mysql_pconnect() establece una conexión a un servidor MySQL. Todos los argumentos son opcionales, y si no existen, se asumen los valores por defecto ('localhost', nombre del usuario propietario del proceso, password vacia).

El hostname puede incluir un número de puerto. ej. "hostname:port" o un camino al socket ej. ":/camino/al/socket" para el puerto para el host local.

> **Nota:** Soporte para ":puerto" fue añadido en 3.0B4.
>
> Soporte para ":/camino/al/socket" fue añadido en 3.0.10.

mysql_pconnect() actua como **mysql_connect()** con dos diferencias fundamentales.

Primero, durante la conexión, la función intenta primero encontrar un enlace persistente abierto con el mismo host, usuario y password. Si lo encuentra, devuelve el identificador de enlace en lugar de abrir otra conexión.

Segundo, la conexión no será cerrada cuando acabe la ejecución del script. El enlace permanecerá abierta para ser usado en el futuro (**mysql_close()** no cierra el enlace establecido con **mysql_pconnect()**).

El parámetro opcional *client_flags* está disponible en PHP 4.3.0.

Este tipo de enlaces son llamados 'persistentes'.

> **Nota:** Note que ese tipo de enlaces solo funcionan si está usando un versión de PHP como módulo. Vea la secciÃ…â€žn Conexiones Persistentes a base de datos para más información.

Aviso
Usando conexiones persistentes puede requerir un poco de ajustar sus configuraciones de Apache y MySQL para asegurarse de que no excede el número de conexiones permitidas por MySQL.

mysql_ping

(PHP 4 >= 4.3.0, PHP 5)

mysql_ping -- Efectuar un chequeo de respuesta (ping) sobre una conexión de servidor o reconectarse si no hay conexión

Descripción

bool **mysql_ping** ([resource id_enlace])

mysql_ping() chequea si está activa o no la conexión con el servidor. Si ésta se ha caído, una reconexión automática es intentada. Esta función puede ser usada por scripts que permanecen pasivos durante largos espacios de tiempo, para chequear si el servidor ha cerrado la conexión, y reconectarse de ser necesario. **mysql_ping()** devuelve **TRUE** si la conexión con el servidor está funcionando, o **FALSE** de lo contrario.

Ejemplo 1. Un ejemplo de mysql_ping()

```
<?php
set_time_limit(0);

$con = mysql_connect('localhost', 'mysqlusuario', 'mypassword');
$bd  = mysql_select_db('mi_bd');

/* Asumiendo que esta consulta toma mucho tiempo */
$resultado = mysql_query($sql);
if (!$resultado) {
    echo 'La consulta #1 fall&oacute;, saliendo.';
    exit;
}

/* Asegurarse de que la conexion sigue viva, si no, intentar una
   re-conexion */
if (!mysql_ping($con)) {
    echo 'Se ha perdido la conexi&oacute;n, saliendo despu&eacute;s de la consulta #1';
    exit;
}
mysql_free_result($result);

/* Ya que la conexion sigue viva, corramos otra consulta */
$resultado2 = mysql_query($sql2);
?>
```

Vea también **mysql_thread_id()** y **mysql_list_processes()**.

mysql_query

(PHP 3, PHP 4 , PHP 5)

mysql_query -- Envía una consulta de MySQL

Descripción

resource **mysql_query** (string query [, resource identificador_de_enlace])

mysql_query() envía una consulta (a la base de datos activa en el servidor asociado con el *identificador_de_enlace* dado).

Lista de parámetros

query

Una consulta SQL

La consulta no debe terminar con punto y coma.

identificador_de_enlace

Un identificador de enlace, como el regresado por **mysql_connect()**.

Si *identificador_de_enlace* no es específicado, se asume el último enlace abierto. Si no se ha abierto enlace, la función intenta establecer uno tal y como si se hubiera llamado a **mysql_connect()** sin argumentos, y lo usará. El resultado de la consulta es puesto en la memoria intermedia (buffer).

Valores retornados

Para las sentencias SELECT, SHOW, DESCRIBE o EXPLAIN, **mysql_query()** regresa un **resource** en caso exitoso, y **FALSE** en error.

Para otro tipo de sentencia SQL, UPDATE, DELETE, DROP, etc, **mysql_query()** regresa **TRUE** en caso exitoso y **FALSE** en error.

El resultado obtenido debe ser pasado a **mysql_fetch_array()**, y otras funciones para el manejo de las tablas del resultado, para accesar los datos regresados.

Use **mysql_num_rows()** para encontrar cuantas filas fueron regresadas para una sentencia SELECT o **mysql_affected_rows()** para encontrar cuantas filas fueron afectadas por una sentencia DELETE, INSERT, REPLACE, o UPDATE.

mysql_query() también fallará y regresará **FALSE** si el usuario no tiene permiso de accesar la o las tablas referenciadas por la consulta.

Ejemplos

Ejemplo 1. Consulta inválida

La siguiente consulta es sintácticamente inválida de tal manera que **mysql_query()** falla y regresa **FALSE**.

```
<?php
$result = mysql_query('SELECT * WHERE 1=1');
if (!$result) {
    die('Invalid query: ' . mysql_error());
}

?>
```

Ejemplo 2. Consulta Válida

La siguiente consulta es válida, así que **mysql_query()** regresa un **resource**.

```php
<?php
// This could be supplied by a user, for example
$firstname = 'fred';
$lastname  = 'fox';

// Formulate Query
// This is the best way to perform a SQL query
// For more examples, see mysql_real_escape_string()
$query = sprintf("SELECT firstname, lastname, address, age FROM friends WHERE firstname
    mysql_real_escape_string($firstname),
    mysql_real_escape_string($lastname));

// Perform Query
$result = mysql_query($query);

// Check result
// This shows the actual query sent to MySQL, and the error. Useful for debugging.
if (!$result) {
    $message  = 'Invalid query: ' . mysql_error() . "\n";
    $message .= 'Whole query: ' . $query;
    die($message);
}

// Use result
// Attempting to print $result won't allow access to information in the resource
// One of the mysql result functions must be used
// See also mysql_result(), mysql_fetch_array(), mysql_fetch_row(), etc.
while ($row = mysql_fetch_assoc($result)) {
    echo $row['firstname'];
    echo $row['lastname'];
    echo $row['address'];
    echo $row['age'];
}

// Free the resources associated with the result set
// This is done automatically at the end of the script
mysql_free_result($result);
?>
```

Ver también

mysql_connect()
mysql_error()
mysql_real_escape_string()
mysql_result()
mysql_fetch_assoc()
mysql_unbuffered_query()

mysql_real_escape_string

(PHP 4 >= 4.3.0, PHP 5)

mysql_real_escape_string -- Escapa caracteres especiales de una cadena para su uso en una sentencia SQL

Descripción

string **mysql_real_escape_string** (string cadena_no_escapada [, resource id_enlace])

cadena_no_escapada

La cadena a escapar

id_enlace (opcional)

El recurso de conexión mysql

Esta función escapará todos los caracteres especiales en la *cadena_no_escapada*, tomando en cuenta el juego de caracteres actual de la conexión, de tal modo que sea seguro usarla con **mysql_query()**. Si desea insertar datos binarios, debe usar ésta función.

mysql_real_escape_string() llama a la función de la biblioteca MySQL mysql_escape_string, la cual coloca barras invertidas antes de los siguientes caracteres: *NULL*, *\x00*, *\n*, *\r*, **, *'*, *"* y *\x1a*.

Ejemplo 1. Ejemplo sencillo de mysql_real_escape_string()

```
<?php
// Conectarse
$enlace = mysql_connect('mysql_host', 'mysql_usuario', 'mysql_contrasenya')
    OR die(mysql_error());

// Consulta
$query = sprintf("SELECT * FROM usuarios WHERE usuario='%s' AND
     password='%s'",
            mysql_real_escape_string($usuario),
            mysql_real_escape_string($password));
?>
```

Es necesario usar siempre (con algunas excepciones) esta función para asegurarse de que los datos sean seguros antes de enviar una consulta a MySQL. Si tiene habilitada la directiva magic_quotes_gpc, y está trabajando con datos que provienen del usuario, debe usar **stripslashes()** primero sobre sus datos. Si sus datos provienen de otras fuentes y tiene habilitada la directiva magic_quotes_runtime, deberá también usar **stripslashes()** con sus datos. De no ser así, quedará abierto a ataques de inyección SQL. He aquí un ejemplo:

Ejemplo 2. Un ejemplo de un ataque de inyección SQL

```
<?php
// Consultar la base de datos para verificar si hay una coincidencia de usuario
$consulta = "SELECT * FROM usuarios WHERE usuario='{$_POST['username']}' AND password='
mysql_query($consulta);

// No revisamos $_POST['password'], podria ser cualquier cosa que el usuario
// quiera! Por ejemplo:
$_POST['username'] = 'aidan';
$_POST['password'] = "' OR 1=1";

// Esto quiere decir que la consulta enviada a MySQL seria:
echo $consulta;
?>
```

La consulta enviada a MySQL:

```
SELECT * FROM usuarios WHERE usuario='aidan' AND password='' OR 1=1
```

¡Esto permitiría que cualquiera iniciara una sesión sin una contraseña válida! Usando **mysql_real_escape_str** previene esto.

```php
<?php
/**
 * Aplicar comillas sobre una variable para hacerla segura
 */
function comillas_inteligentes($valor)
{
    // Retirar las barras si es necesario
    if (get_magic_quotes_gpc()) {
        $valor = stripslashes($valor);
    }

    // Colocar comillas si no es un entero
    if (!is_int($valor)) {
        $valor = "'" . mysql_real_escape_string($valor) . "'";
    }

    return $valor;
}

// Conexion
$enlace = mysql_connect('mysql_host', 'mysql_usuario', 'mysql_contrasenya')
    OR die(mysql_error());

// Realizar una consulta segura
$consulta = sprintf("SELECT * FROM usuarios WHERE usuario=%s AND password=%s",
            comillas_inteligentes($_POST['username']),
            comillas_inteligentes($_POST['password']));

mysql_query($consulta);
?>
```

La consulta será ejecutada correctamente ahora, y los ataques de inyección no darán resultado.

> **Nota: mysql_real_escape_string()** no escapa *%* ni _. Éstos son comodines en MySQL si se combianan con *LIKE*, *GRANT*, o *REVOKE*.

Vea también **mysql_client_encoding()**, **addslashes()**, **stripslashes()**, y las directivas magic_quotes_gpc y magic_quotes_runtime.

mysql_result

(PHP 3, PHP 4 , PHP 5)

mysql_result -- Devuelve datos de un resultado

Descripción

int **mysql_result** (int id_resultado, int numero_de_fila [, mixed campo])

mysql_result() devuelve el contenido de una celda de un resultado MySQL. El campo argumento puede ser el nombre del campo o el offset o tabla.nombre_del_campo. Si el nombre de la columna tiene un alias ('select foo as bar from...'), utilice el alias en lugar del nombre de la columna.

Cuando se trabaja un un gran resultado, debe considerarse la utilización de una función que devuelva una fila entera ya que estas funciones son MUCHO más rápidas que **mysql_result()**. También, especificando un offset numérico en lugar del nombre del campo, la ejecución será más rápida.

Las llamadas a **mysql_result()** no deben mezclarse con llamadas a las otras sentencias que trabajan con un identificador de resultado.

Ejemplo 1. Ejemplo mysql_result

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
    die('Could not connect: ' . mysql_error());
}
$result = mysql_query('SELECT name FROM work.employee');
if (!$result) {
    die('Could not query:' . mysql_error());
}
echo mysql_result($result, 2); // outputs third employee's name

mysql_close($link);
?>
```

Alternativas recomendadas: **mysql_fetch_row()**, **mysql_fetch_array()**, **mysql_fetch_assoc()**, **mysql_fetch_object()**.

mysql_select_db

(PHP 3, PHP 4 , PHP 5)

mysql_select_db -- Selecciona un base de datos MySQL

Descripción

int **mysql_select_db** (cadena base_de_datos [, int identificador_de_enlace])

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

mysql_select_db() establece la base activa que estará asociada con el identificador de enlace específicado. Si no se específica un identificador de enlace, se asume el último enlace abierto. Si no hay ningún enlace abierto, la función intentará establecer un enlace como si se llamara a **mysql_connect()**.

Toda llamada posterior a **mysql_query()** utilizará la base activada.

Ejemplo 1. Ejemplo mysql_select_db

```
<?php

$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
    die('Not connected : ' . mysql_error());
}

// make foo the current db
$db_selected = mysql_select_db('foo', $link);
if (!$db_selected) {
    die ('Can\'t use foo : ' . mysql_error());
}
?>
```

Ver también: **mysql_connect()**, **mysql_pconnect()**, **mysql_query()**.

Por razones de compatibilidad puede usarse también **mysql_selectdb()**. Sin embargo esta función es obsoleta.

mysql_stat

(PHP 4 >= 4.3.0, PHP 5)

mysql_stat -- Obtener el status actual del sistema

Descripción

string **mysql_stat** ([resource id_enlace])

mysql_stat() devuelve el status actual del servidor.

> **Nota:** Por el momento, **mysql_stat()** devuelve únicamente el status de uptime, hilos, consultas, tablas abiertas, tablas de vaciado y consultas por segundo. Para una lista completa de variables de status, usted tendrá que usar el comando SQL 'SHOW STATUS'.

Ejemplo 1. Ejemplo de mysql_stat()

```
<?php
$enlace = mysql_connect('localhost', "mysql_user", "mysql_password");
$status = explode('  ', mysql_stat($enlace));
print_r($status);
?>
```

El anterior ejemplo produciría la siguiente salida:

```
Array
(
    [0] => Uptime: 5380
    [1] => Threads: 2
    [2] => Questions: 1321299
    [3] => Slow queries: 0
    [4] => Opens: 26
    [5] => Flush tables: 1
    [6] => Open tables: 17
    [7] => Queries per second avg: 245.595
)
```

Vea también **mysql_get_server_info()** y **mysql_list_processes()**.

mysql_tablename

(PHP 3, PHP 4 , PHP 5)

mysql_tablename -- Devuelve el nombre de la tabla de un campo

Descripción

cadena **mysql_tablename** (int id_resultado, int i)

mysql_tablename() toma un puntero de resultado devuelto por **mysql_list_tables()** así como un índice (integer) y devuelve el nombre de una tabla. Se puede usar la función **mysql_num_rows()** para determinar el número de tablas en el puntero de resultado. Use la función **mysql_tablename()** para usar este apuntador de resultado, o cualquier función para las tablas resultantes como **mysql_fetch_array()**.

Ejemplo 1. Ejemplo mysql_tablename

```
<?php
mysql_connect("localhost", "mysql_user", "mysql_password");
$result = mysql_list_tables("mydb");

$num_rows = mysql_num_rows($result);
for ($i = 0; $i < $num_rows; $i++) {
    echo "Table: ", mysql_tablename($result, $i), "\n";
}

mysql_free_result($result);
?>
```

Vea también **mysql_list_tables()**, **mysql_field_table()**, **mysql_db_name()**.

mysql_thread_id

(PHP 4 >= 4.3.0, PHP 5)

mysql_thread_id -- Devuelve el ID del hilo actual

Descripción

int **mysql_thread_id** ([resource id_enlace])

mysql_thread_id() devuelve el ID del hilo actual. Si la conexión se ha perdido y se reconecta mediante **mysql_ping()**, el ID del hilo cambiará. Esto quiere decir que usted no debería obtener el ID del hilo actual y almacenarlo para después. Es mejor obtenerlo cuando lo necesita.

Ejemplo 1. Ejemplo de mysql_thread_id()

```
<?php
$enlace = mysql_connect('localhost', 'mysql_user', 'mysql_password');
$id_hilo = mysql_thread_id($enlace);
if ($id_hilo){
    printf("El ID del hilo actual es %d\n", $id_hilo);
}
?>
```

El anterior ejemplo produciría la siguiente salida:

```
El ID del hilo actual es 73
```

Vea también **mysql_ping()** y **mysql_list_processes()**.

mysql_unbuffered_query

(PHP 4 >= 4.0.6, PHP 5)

mysql_unbuffered_query -- Envía una consulta SQL a MySQL, sin recuperar ni colocar en búfer las filas de resultado

Descripción

resource **mysql_unbuffered_query** (string consulta [, resource id_enlace])

mysql_unbuffered_query() envía la *consulta* SQL a MySQL, sin recuperar ni colocar en búfer las filas de resultado automáticamente, como **mysql_query()** lo hace. Por una parte, esto ahorra una considerable cantidad de memoria con las consultas SQL que producen conjuntos grandes de resultados. Por otra parte, usted puede empezar a trabajar con el conjunto de resultado

inmediatamente después de que la primera fila ha sido recuperada: no necesita esperar hasta que la consulta SQL completa haya sido ejecutada. Cuando se usan múltiples conexiones con la BD, necesita indicar el parámetro opcional *id_enlace*.

> **Nota:** Los beneficios de **mysql_unbuffered_query()** vienen por un precio: No puede usar **mysql_num_rows()** ni **mysql_data_seek()** en un conjunto de resultados devuelto por **mysql_unbuffered_query()**. También tendrá que recuperar todas las filas de resultado de una consulta SQL sin búfer, antes de poder enviar una nueva consulta SQL a MySQL.

Vea también **mysql_query()**.

LXXX. Extensión mejorada de MySQL

Introducción

La extensión mysqli permite acceder a la funcionalidad proveida por MySQL 4.1 y superior. Más información acerca del servidor de base de datos MySQL puede ser encontrada en http://www.mysql.com/

La documentación para MySQL puede ser encontrada en http://dev.mysql.com/doc/.

Partes de esta documentación ha sido incluida del manual de MySQL con permiso de MySQL AB.

Requirimientos

Para tener estas funciones disponibles, usted debe compilar PHP con soporte para la extensión MySQLI.

> **Nota:** La extensión mysqli está diseñada para trabajar con la versión 4.1.3 o superior de MySQL. Para versiones previas, por favor vea documentación de la extensión MySQL.

Instalación

Para instalar la extensión de mysqli para PHP, use la opción de configuración *--with-mysqli=mysql_config_path/mysql_config* donde *mysql_config_path* representa la ubicación del programa *mysql_config* que viene con MySQL en versiones superiores a 4.1.

Si quiere instalar la extensión de mysql junto con la extensión mysqli, usted tiene que usar la mismas librerias cliente, para evitar cualquier conflicto.

Configuración en tiempo de ejecución

El comportamiento de estas funciones está afectado por los valores definidos en `php.ini`.

Tabla 1. Opciones de configuración de MySQLi

Nombre	Default	Cambiable
mysqli.max_links	"-1"	PHP_INI_SYSTEM
mysqli.default_port	NULL	PHP_INI_ALL
mysqli.default_socket	NULL	PHP_INI_ALL
mysqli.default_host	NULL	PHP_INI_ALL
mysqli.default_user	NULL	PHP_INI_ALL
mysqli.default_pw	NULL	PHP_INI_ALL

Para más detalles y definiciones de las constantes PHP_INI_* anteriores, vea el capítulo sobre Cambios de configuración.

A continuación se presenta una corta explicación de las directivas de configuración.

mysqli.max_links **integer**

El número máximo de conexiones MySQL por proceso.

mysqli.default_port **string**

El puerto TCP por defecto a usar cuando se conecte al servidor de base de datos si no se ha específicado otro puerto. Si no se define el default, el puerto será tomado de la variable de ambiente *MYSQL_TCP_PORT*, la entrada *mysql-tcp* en el archivo `/etc/services` o la constante en tiempo de ejecución *MYSQL_PORT* en ese orden. Win32 usará solo la constante *MYSQL_PORT*.

mysqli.default_socket **string**

El nombre de socket por defecto a usar cuando se conecte a un servidor de base de datos local si no se ha específicado otro nombre de socket.

mysqli.default_host **string**

El servidor huésped por defecto a usar cuando se conecta al servidor de base de datos si no se ha específicado otro. No aplica en modo seguro.

mysqli.default_user **string**

El nombre de usuario por defecto a usar cuando se conecta al servidor de base de datos si no se específica otro nombre. No aplica en modo seguro.

mysqli.default_pw **string**

La contraseña a usar cuando se conecta al servidor de base de datos si no se específica otra contraseña. No aplica en modo seguro.

Clases predefinidas

mysqli

Representa una conexión entres PHP y la base de datos MySQL.

Constructor

- mysqli - constructor de un nuevo objeto mysqli

Métodos

- autocommit - cambia en on o off las modificaciones de auto-commiting en la base de datos
- change_user - cambia el usuario del identificador de enlace específicado
- character_set_name - regresa el conjunto de caracteres del identificador de enlace específicado
- close - cierra una conexión previamente abierta
- commit - completa la transacció actual
- connect - abre una nueva conexión al servidor de MySQL
- debug - realiza operaciones de rastreo
- dump_debug_info - vacía informació de rastreo
- get_client_info - regresa la versión del cliente
- get_host_info - regresa el tipo de conexión usada
- get_server_info - regresa la versión del servidor MySQL
- get_server_version - regresa la versión del servidor MySQL
- init - inicializa el objeto mysqli
- info - obtiene información acerca de la consulta más recientemente ejecutada
- kill - solicita al servidor destruir un thread de mysql
- multi_query - realiza consultas de SQL múltiples
- more_results - checa si existen mas resultados de la consulta múltiple actualmente ejecutada
- next_result - lee el siguiente resultado de la consulta múltiple actualmente ejecutada
- options - fija opciones

- ping - llama una conexión con el servidor o reconecta si no hay conexión
- prepare - prepara una sentencia SQL
- query - ejecuta una sentencia SQL
- real_connect - intenta abrir una conexión al servidor de base de datos MySQL
- escape_string - caracteres especiales de escape para ser usados en una sentencia SQL, tomando en cuenta el conjunto de caracteres actual de la conexión
- rollback - deshace la transacción actual
- select_db - selecciona la base de datos por defecto
- ssl_set - fija los parámetros SSL
- stat - obtiene el estado actual del sistema
- stmt_init - inicializa una sentencia para ser usada con mysqli_stmt_prepare
- store_result - transfiere un resultado de la última consulta
- use_result - transfiere un resultado sin almacenamiento intermedio de la última consulta
- thread_safe - regresa si se ha dado thread_safety o no

Propiedades

- affected_rows - obtiene el número de filas afectadas en la operación MySQL previa
- client_info - regresa en una cadena la versión del cliente de MySQL
- client_version - regresa en un entero la versión del cliente de MySQL
- errno - regresa el código de error para la llamada a función más reciente
- error - regresa la cadena de error para la llamada a función más reciente
- field_count - regresa el número de columnas para la consulta más reciente
- host_info - regresa la cadena representando el tipo de conexión usado
- info - regresa información acerca de la consulta ejecutada más recientemente
- insert_id - regresa el identificador generado automáticamente usado en la última consulta
- protocol_version - regresa la versión del protocolo usado por MySQL
- sqlstate - regresa una cadena que contiene el código de error SQLSTATE para el último error

- thread_id - regresa el identificador del THREAD para la conexión actual
- warning_count - regresa el número de alertas generadas durante la ejecución del enunciado SQL previo

---

mysqli_stmt

Representa una sentencia preparada.

Métodos

- bind_param - enlaza variables a una sentencia preparada
- bind_result - enlaza variables a una sentencia preparada para el almacenamiento del resultado
- close - cierra la sentencia preparada
- data_seek - busca a una fila arbitrariamente en el resultado
- execute - ejecuta una sentencia preparada
- fetch - obtiene el resultado de la sentencia preparada en variables límite
- free_result - libera la memoria del resultado almacenado para el manejador de la sentencia dada
- result_metadata - obtiene un resultado de la sentencia preparada para información de metadatos
- prepare - prepara una consulta SQL
- send_long_data - envía datos en partes
- reset - resetea una sentencia preparada
- store_result - almacena el resultado completo de la sentencia preparada

---

Propiedades

- affected_rows - regresa las filas afectadas de la última sentencia ejecutdada
- errno - regresa el código de error para la llamada a función más reciente
- errno - regresa el mensaje de error para la llamada a función más reciente
- param_count - regresa el numero de parámetros para la sentencia preparada dada
- sqlstate - regresa una cadena conteniendo el código de error SQLSTATE para la llamada a

función más reciente

---

mysqli_result

Representa el resultado obtenido de la consulta hecha en la base de datos.

Métodos

- close - cierra el resultado
- data_seek - mueve el puntero interno del resultado
- fetch_field - obtiene la información de la columna en el resultado
- fetch_fields - obtiene la información para todas las columnas del resultado
- fetch_field_direct - obtiene información de la columna para la columna dada
- fetch_array - recupera una fila como una matriz asociativa, una matriz numérica, o ambos.
- fetch_assoc - obtiene una fuka como una matriz asociativa
- fetch_object - obtiene una fila como un objeto
- fetch_row - obtiene una fila como una matriz enumerada
- close - libera la memoria ocupada por el resultado
- field_seek - fija el apuntador del resultado a la posición específicada

---

Propiedades

- current_field - regresa la posición de puntero actual
- field_count - regresa el número de campos en el resultado
- lengths - regresa una matriz con la longitud de los campos
- num_rows - regresa el número de filas en el resultado

Constantes predefinidas

Tabla 2. Constantes MySQLi

Nombre	Descripción
MYSQLI_READ_DEFAULT_GROUP (integer)	Lee las opciones del grupo llamado de `my.cnf' o la fila especificada con *MYSQLI_READ_DEFAULT_FILE*
MYSQLI_READ_DEFAULT_FILE (integer)	Lee las opciones del archivo específicado en vez de *my.cnf*
MYSQLI_OPT_CONNECT_TIMEOUT (integer)	Tiempo de espera para la conexión en segundos
MYSQLI_OPT_LOCAL_INFILE (integer)	Habilita el comando *LOAD LOCAL INFILE*
MYSQLI_INIT_COMMAND (integer)	Comando a ejecutar cuando se conecta al servidor MySQL. Será automáticamente re-ejecutado cuando se vuelva a conectar.
MYSQLI_CLIENT_SSL (integer)	Usa SSL (protocolo de encripción). Esta opción no debe ser fijado por programas de apliación; esta es fijada internamente en la libreria cliente de MySQL
MYSQLI_CLIENT_COMPRESS (integer)	Usa el protocolo de compresión
MYSQLI_CLIENT_INTERACTIVE (integer)	Permite tiempo de espera interactivo en segundos (en vez de esperar ciertos segundos) de inactividad antes de cerrar la conexión. La variable wait_timeout de sesión del cliente será fijada a el valor de la variable de sesión interactive_timeout.
MYSQLI_CLIENT_IGNORE_SPACE (integer)	Permite espacios después de los nombres de las funciones. Esto hace a todas las funciones palabras reservadas.
MYSQLI_CLIENT_NO_SCHEMA (integer)	No permite la sintaxis db_name.tbl_name.col_name syntax.
MYSQLI_CLIENT_MULTI_QUERIES (integer)	
MYSQLI_STORE_RESULT (integer)	Para usar resultados con almacenamiento intermedio
MYSQLI_USE_RESULT (integer)	Para usar resultados sin almacenamiento intermedio
MYSQLI_ASSOC (integer)	Las columnas son regresadas en una matriz que tiene el nombre del campo como índice de la matriz.
MYSQLI_NUM (integer)	Las columnas son regresadas en una matriz teniendo un índice enumerado.
MYSQLI_BOTH (integer)	Las clumnas son regresadas en una matriz teniendo ambas formas de índice numérico y con el nombre del campo como índice asociativo.
MYSQLI_NOT_NULL_FLAG (integer)	Indica que un campo es definido como *NOT NULL*
MYSQLI_PRI_KEY_FLAG (integer)	El campo es parte de un índice primario
MYSQLI_UNIQUE_KEY_FLAG (integer)	El campo es parte de un índice único
MYSQLI_MULTIPLE_KEY_FLAG (integer)	El campo es parte de un índice.

Nombre	Descripción
MYSQLI_BLOB_FLAG (integer)	El campo está definido como *BLOB*
MYSQLI_UNSIGNED_FLAG (integer)	El campo está definido como *UNSIGNED*
MYSQLI_ZEROFILL_FLAG (integer)	El campo está definido como *ZEROFILL*
MYSQLI_AUTO_INCREMENT_FLAG (integer)	El campo está definido como *AUTO_INCREMENT*
MYSQLI_TIMESTAMP_FLAG (integer)	El campo está definido como *TIMESTAMP*
MYSQLI_SET_FLAG (integer)	El campo está definido como *SET*
MYSQLI_NUM_FLAG (integer)	El campo está definido como *NUMERIC*
MYSQLI_PART_KEY_FLAG (integer)	El campo es parte de un índice multiple
MYSQLI_GROUP_FLAG (integer)	El campo es parte de *GROUP BY*
MYSQLI_TYPE_DECIMAL (integer)	El campo está definido como *DECIMAL*
MYSQLI_TYPE_TINY (integer)	El campo está definido como *TINYINT*
MYSQLI_TYPE_SHORT (integer)	El campo está definido como *INT*
MYSQLI_TYPE_LONG (integer)	El campo está definido como *INT*
MYSQLI_TYPE_FLOAT (integer)	El campo está definido como *FLOAT*
MYSQLI_TYPE_DOUBLE (integer)	El campo está definido como *DOUBLE*
MYSQLI_TYPE_NULL (integer)	El campo está definido como *DEFAULT NULL*
MYSQLI_TYPE_TIMESTAMP (integer)	El campo está definido como *TIMESTAMP*
MYSQLI_TYPE_LONGLONG (integer)	El campo está definido como *BIGINT*
MYSQLI_TYPE_INT24 (integer)	El campo está definido como *MEDIUMINT*
MYSQLI_TYPE_DATE (integer)	El campo está definido como *DATE*
MYSQLI_TYPE_TIME (integer)	El campo está definido como *TIME*

Nombre	Descripción
MYSQLI_TYPE_DATETIME (integer)	El campo está definido como *DATETIME*
MYSQLI_TYPE_YEAR (integer)	El campo está definido como *YEAR*
MYSQLI_TYPE_NEWDATE (integer)	El campo está definido como *DATE*
MYSQLI_TYPE_ENUM (integer)	El campo está definido como *ENUM*
MYSQLI_TYPE_SET (integer)	El campo está definido como *SET*
MYSQLI_TYPE_TINY_BLOB (integer)	El campo está definido como *TINYBLOB*
MYSQLI_TYPE_MEDIUM_BLOB (integer)	El campo está definido como *MEDIUMBLOB*
MYSQLI_TYPE_LONG_BLOB (integer)	El campo está definido como *LONGBLOB*
MYSQLI_TYPE_BLOB (integer)	El campo está definido como *BLOB*
MYSQLI_TYPE_VAR_STRING (integer)	El campo está definido como *VARCHAR*
MYSQLI_TYPE_STRING (integer)	El campo está definido como *CHAR*
MYSQLI_TYPE_GEOMETRY (integer)	El campo está definido como *GEOMETRY*
MYSQLI_NEED_DATA (integer)	Obtiene más información para varibles enlazadas (bind)
MYSQLI_NO_DATA (integer)	No obtiene más información para variables enlazadas (bind)

Ejemplos

Todos los ejemplos en la documentación MySQLI usan la base de datos de MySQL AB. Puede ser encontrada en http://dev.mysql.com/get/Downloads/Manual/world.sql.gz/from/pick

Tabla de contenidos

mysqli_connect_error -- Regresa una descripción del último error de la conexión
mysqli_connect -- Abre una nueva conexión al servidor MySQL
mysqli_data_seek -- Ajusta el apuntador arbitrariamente a una fila en el resultado
mysqli_debug -- Realiza operaciones de rastreo de errores
mysqli_disable_reads_from_master -- Deshabilita lectura del Servidor maestro
mysqli_disable_rpl_parse -- Deshabilita el analizador RPL
mysqli_dump_debug_info -- Vacía información de rastreo de errores en el log
mysqli_embedded_connect -- Abre una conexió a un servidor MySQL embebido
mysqli_enable_reads_from_master -- Habilita la lectura del servidor Maestro
mysqli_enable_rpl_parse -- Habilita el analizador RPL
mysqli_errno -- Regresa el código de error para la función más recientemente llamada
mysqli_error -- Regresa una cadena con la descripció del último error
mysqli_escape_string -- Alias de **mysqli_real_escape_string()**
mysqli_execute -- Alias de **mysqli_stmt_execute()**
mysqli_fetch_array -- Obtiene una fila como una matriz asociativa, una matriz numérica o ambos
mysqli_fetch_assoc -- Obtiene una fila del resultado como una matriz asociativa
mysqli_fetch_field_direct -- Obtiene los metadatos de un campo
mysqli_fetch_field -- Regresa metadatos de el campo en el resultado
mysqli_fetch_fields -- Regresa una matriz de objetos representando los campos en un resultado
mysqli_fetch_lengths -- Regresa la longitud de las columnas de la fila actual en el resultado
mysqli_fetch_object -- Regresa la fila actual del resultado como un objeto
mysqli_fetch_row -- Obtiene una fila del resultado como una matriz enumerada
mysqli_fetch -- Alias de **mysqli_stmt_fetch()**
mysqli_field_count -- Regresa el número de columnas para la consulta más reciente
mysqli_field_seek -- Fija el apuntador del resultado a una posicion específica
mysqli_field_tell -- Obtiene la posición de desplazamiento del apuntador de un resultado
mysqli_free_result -- Libera la memoria asociada con el resultado
mysqli_get_client_info -- Regresa en una cadena la versión del cliente de MySQL
mysqli_get_client_version -- Obtiene información del cliente MySQL
mysqli_get_host_info -- Regresa una cadena que representa el tipo de conexión usada
mysqli_get_metadata -- Alias para **mysqli_stmt_result_metadata()**
mysqli_get_proto_info -- Regresa la versión del protocolo de MySQL usado
mysqli_get_server_info -- Regresa la version del servidor de MySQL
mysqli_get_server_version -- Regresa la versión del servidor MySQL como un entero
mysqli_info -- Obtiene información acerca de la consulta más recientemente ejecutada
mysqli_init -- Inicializa MySQLi y regresa un objeto para ser usado con mysqli_rea_connect
mysqli_insert_id -- Regresa el ID generado automáticamente en la última consulta
mysqli_kill -- Le pide al servidor "matar" el proceso MySQL
mysqli_master_query -- Hace cumplir la ejecución de una consulta en el servidor en una configuración cliente/servidor
mysqli_more_results -- Checa si hay má resultados de consultas de una consulta múltiple
mysqli_multi_query -- Ejecuta una consulta en la base de datos
mysqli_next_result -- Prepara el siguiente resultado de una consulta múltiple
mysqli_num_fields -- Obtiene el número de campos en un resultado Get the number of fields in a result
mysqli_num_rows -- Obtiene el número de filas en un resultado
mysqli_options -- Fija opciones
mysqli_param_count -- Alias para **mysqli_stmt_param_count()**
mysqli_ping -- Revisa una conexión al servidor o intenta reconectar si la conexión se ha perdido
mysqli_prepare -- Prepara una sentencia SQL para su ejecución
mysqli_query -- Ejecuta una consulta en la base de datos
mysqli_real_connect -- Abre una conexión a un servidor de MySQL
mysqli_real_escape_string -- Protege caracteres especiales en una cadena para ser usada en una sentencia SQL, tomando en cuenta el conjunto de caracteres para la conexión
mysqli_real_query -- Ejecuta una consulta SQL

mysqli_report -- Habilita o deshabilita las funciones internas de reporte
mysqli_rollback -- Regresa/deshace la transacción actual
mysqli_rpl_parse_enabled -- Revisa si el analizador RPL está habilitado
mysqli_rpl_probe -- Prueba RPL
mysqli_rpl_query_type -- Regresa el tipo de consulta RPL
mysqli_select_db -- Selecciona la base de datos por defecto para las consultas de la base de datos
mysqli_send_long_data -- Alias para **mysqli_stmt_send_long_data()**
mysqli_send_query -- Envía la consulta y regresa
mysqli_server_end -- Termina el proceso del Servidor
mysqli_server_init -- Inicializa un servidor embebido
mysqli_set_opt -- Alias de **mysqli_options()**
mysqli_sqlstate -- Regresa el error SQLSTATE de la operación MySQL previa
mysqli_ssl_set -- Usado para establecer conexiones seguras usando SSL
mysqli_stat -- Obtiene el estado actual del sistema
mysqli_stmt_affected_rows -- Regresa el número total de filas cambiadas, borradas o insertadas por la última sentencia ejecutada
mysqli_stmt_bind_param -- Enlaza variables como parámetros a una sentencia preparada
mysqli_stmt_bind_result -- Enlaza variables a una sentencia preparada para el almacenamiento del resultado
mysqli_stmt_close -- Cierra la sentencia preparada
mysqli_stmt_data_seek -- Busca una fila arbitrariamente en el resultado
mysqli_stmt_errno -- Regresa el código de error para la llamada más reciente
mysqli_stmt_error -- Regresa una descipción para el último error
mysqli_stmt_execute -- Ejecuta una consulta preparada
mysqli_stmt_fetch -- Obtiene el resultado de una sentencia SQL preparada en las variables enlazadas
mysqli_stmt_free_result -- Libera la memoria donde se almacenó el resultado
mysqli_stmt_init -- Inicializa una sentencia y regresa un objeto para ser usado con mysqli_stmt_prepare
mysqli_stmt_num_rows -- Regresa el número de filas en un resultado de la sentencia
mysqli_stmt_param_count -- Regresa el número de parámetros para la sentencia dada
mysqli_stmt_prepare -- Prepara una sentencia SQL para su ejecución
mysqli_stmt_reset -- Resetea una sentencia preparada
mysqli_stmt_result_metadata -- Regresa metadatos del resultado de una sentencia preparada
mysqli_stmt_send_long_data -- Envía los datos en bloques
mysqli_stmt_sqlstate -- Regresa el error SQLSTATE de la operación con sentencias previa
mysqli_stmt_store_result -- Transfiere un resultado de una sentencia preparada
mysqli_store_result -- Transfiere un resultado de la última consulta
mysqli_thread_id -- Regresa el identificador del THREAD para la conexión actual
mysqli_thread_safe -- Regresa si se ha dado un THREAD seguro o no Returns whether thread safety is given or not
mysqli_use_result -- Inicia una recuperación de un resultado
mysqli_warning_count -- Regresa el número de alertas de la última consulta para el identificador de enlace dado

mysqli_affected_rows

(PHP 5)

mysqli_affected_rows

(no version information, might be only in CVS)

mysqli->affected_rows -- Obtiene el número de filas afectadas en una operación de MySQL previa

Descripción

Estilo por procedimientos:

mixto **mysqli_affected_rows** (mysqli identificador_de_enlace)

Estilo orientado a objetos (característica):

class **mysqli** {

mixto affected_rows

}

mysqli_affected_rows() Regresa el número de filas afectadas por la última consulta INSERT, UPDATE, o DELETE asociada con el *identificador_de_enlace* dado. Si la última consulta fue invalida, esta función regresará -1.

> **Nota:** Para sentencias SELECT **mysqli_affected_rows()** trabaja igual a **mysqli_num_rows()**.

La función **mysqli_affected_rows()** sólo trabaja con consultas que modifican o afectan una tabla. En caso de que necesite el número de filas de una consulta SELECT, use la función **mysqli_num_rows()** en su lugar.

Valores retornados

Un entero mayor a cero indica el número de filas afectadas u obtenidas. Cero indica que no se actualizaron registros para una sentencia UPDATE, no hubo coincidencias con la clausula WHERE en la consulta o que no se ha ejecutado aún ninguna consulta. -1 indica que la consulta regreso un error.

> **Nota:** Si el número de filas afectadas es mayor que el valor entero máximo, entonces el número de filas afectadas será regresado como una cadena.

Ver también

mysqli_num_rows(), y **mysqli_info()**.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```php
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

/* Insert rows */
$mysqli->query("CREATE TABLE Language SELECT * from CountryLanguage");
printf("Affected rows (INSERT): %d\n", $mysqli->affected_rows);

$mysqli->query("ALTER TABLE Language ADD Status int default 0");

/* update rows */
$mysqli->query("UPDATE Language SET Status=1 WHERE Percentage > 50");
printf("Affected rows (UPDATE): %d\n", $mysqli->affected_rows);

/* delete rows */
$mysqli->query("DELETE FROM Language WHERE Percentage < 50");
printf("Affected rows (DELETE): %d\n", $mysqli->affected_rows);

/* select all rows */
$result = $mysqli->query("SELECT CountryCode FROM Language");
printf("Affected rows (SELECT): %d\n", $mysqli->affected_rows);

$result->close();

/* Delete table Language */
$mysqli->query("DROP TABLE Language");

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

if (!$link) {
    printf("Can't connect to localhost. Error: %s\n", mysqli_connect_error());
    exit();
}

/* Insert rows */
mysqli_query($link, "CREATE TABLE Language SELECT * from CountryLanguage");
printf("Affected rows (INSERT): %d\n", mysqli_affected_rows($link));

mysqli_query($link, "ALTER TABLE Language ADD Status int default 0");

/* update rows */
mysqli_query($link, "UPDATE Language SET Status=1 WHERE Percentage > 50");
printf("Affected rows (UPDATE): %d\n", mysqli_affected_rows($link));

/* delete rows */
mysqli_query($link, "DELETE FROM Language WHERE Percentage < 50");
printf("Affected rows (DELETE): %d\n", mysqli_affected_rows($link));

/* select all rows */
$result = mysqli_query($link, "SELECT CountryCode FROM Language");
printf("Affected rows (SELECT): %d\n", mysqli_affected_rows($link));

mysqli_free_result($result);

/* Delete table Language */
mysqli_query($link, "DROP TABLE Language");

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Affected rows (INSERT): 984
Affected rows (UPDATE): 168
Affected rows (DELETE): 815
Affected rows (SELECT): 169
```

mysqli_autocommit

(PHP 5)

mysqli_autocommit

(no version information, might be only in CVS)

mysqli->autocommit -- Activa o desactiva la modificación de auto-entrega de la base de datos

Descripción

Estilo por procedimientos:

bool **mysqli_autocommit** (mysqli identificador_de_enlace, bool modo)

Estilo orientado a objetos (método)

class **mysqli** {

bool **autocommit** (bool modo)

}

mysqli_autocommit() Es usado para cambiar entre on u off el modo de auto-entrega en las consultas para la conexión a la base de datos representada por el objeto *identificador_de_enlace*.

> **Nota: mysqli_autocommit()** No funciona sobre tablas que no son del tipo transaccional (como MyISAM o ISAM).
>
> Para determinar el estado actual de auto-entrega use el comando SQL: 'SELECT @@autocommit'.

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Ver también

mysqli_commit(), y **mysqli_rollback()**.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

/* turn autocommit on */
$mysqli->autocommit(TRUE);

if ($result = $mysqli->query("SELECT @@autocommit")) {
    $row = $result->fetch_row();
    printf("Autocommit is %s\n", $row[0]);
    $result->free();
}

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

if (!$link) {
    printf("Can't connect to localhost. Error: %s\n", mysqli_connect_error());
    exit();
}

/* turn autocommit on */
mysqli_autocommit($link, TRUE);

if ($result = mysqli_query($link, "SELECT @@autocommit")) {
    $row = mysqli_fetch_row($result);
    printf("Autocommit is %s\n", $row[0]);
    mysqli_free_result($result);
}

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Autocommit is 1
```

mysqli_bind_param

mysqli_bind_param -- Alias para **mysqli_stmt_bind_param()**

Descripción

Esta función es un alias de **mysqli_stmt_bind_param()**. Para una descripción detallada vea **mysqli_stmt_bind_param()**.

> **Nota: mysqli_bind_param()** es obsoleta y será removida.

Ver también

mysqli_stmt_bind_param().

mysqli_bind_result

mysqli_bind_result -- Alias para **mysqli_stmt_bind_result()**

Descripción

Esta funció es un alias de **mysqli_stmt_bind_result()**. Para una descripción detallada vea **mysqli_stmt_bind_result()**.

> **Nota: mysqli_bind_result()** es obsoleta y será removida.

Ver también

mysqli_stmt_bind_result().

mysqli_change_user

(PHP 5)

mysqli_change_user

(no version information, might be only in CVS)

mysqli->change_user -- Cambia el usuario de la conexión a la base de datos especificada

Descripción

Estilo por procedimientos:

bool **mysqli_change_user** (mysqli identificador_de_enlace, cadena usuario, cadena contraseña, cadena base_de_datos)

Estilo orientado a objetos (método):

class **mysqli** {

bool **change_user** (cadena usuario, cadena contraseÃ…â€ža, cadena base_de_datos)

}

mysqli_change_user() Es usada para cambiar el usuario para la conexión de base de datos indicada por el parámetro *identificador_de_enlace* y para fijar la actual base de datos a la específicada por el parámetro *base_de_datos*.

Si se desea, se puede pasar el valor **`NULL`** en lugar del parámetro *base_de_datos* resultando en solo cambiar el usaurio y no seleccionar la base de datos. Para seleccionar una base de datos en este caso use la función **mysqli_select_db()**.

Para poder cambiar de usuario de forma exitosa se debe proveer de un *usuario* y *contraseña* validos, y tales parámetros debe contar con los suficientes permisos para acceder a la base de datos deseada. Si por cualquier razón la autorización falla, el actual usuario autenticado permanecerá activo.

> **Nota:** El uso de este comando siempre producirá que la conexión actual a la base de datos se comporte como si fuera una nueva conexión, sin importar si la operación fue completada exitosamente. Este reinicio implica el hacer la restauración no actualizada "rollback" de cualquier transacción activa, cerrar todas las tablas temporales y des-asegurar todas las tablas aseguradas.

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Ver también

mysqli_connect(), y **mysqli_select_db()**.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php

/* connect database test */
$mysqli = new mysqli("localhost", "my_user", "my_password", "test");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

/* Set Variable a */
$mysqli->query("SET @a:=1");

/* reset all and select a new database */
$mysqli->change_user("my_user", "my_password", "world");

if ($result = $mysqli->query("SELECT DATABASE()")) {
    $row = $result->fetch_row();
    printf("Default database: %s\n", $row[0]);
    $result->close();
}

if ($result = $mysqli->query("SELECT @a")) {
    $row = $result->fetch_row();
          if ($row[0] === NULL) {
            printf("Value of variable a is NULL\n");
          }
    $result->close();
}

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
/* connect database test */
$link = mysqli_connect("localhost", "my_user", "my_password", "test");

/* check connection */
if (!$link) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

/* Set Variable a */
mysqli_query($link, "SET @a:=1");

/* reset all and select a new database */
mysqli_change_user($link, "my_user", "my_password", "world");

if ($result = mysqli_query($link, "SELECT DATABASE()")) {
    $row = mysqli_fetch_row($result);
    printf("Default database: %s\n", $row[0]);
    mysqli_free_result($result);
}

if ($result = mysqli_query($link, "SELECT @a")) {
    $row = mysqli_fetch_row($result);
    if ($row[0] === NULL) {
            printf("Value of variable a is NULL\n");
          }
    mysqli_free_result($result);
}

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Default database: world
Value of variable a is NULL
```

mysqli_character_set_name

(PHP 5)

mysqli_character_set_name

(no version information, might be only in CVS)

mysqli->character_set_name -- Regresa el conjunto de caracteres determinados por default para la conexión de base de datos

Descripción

Estilo por procedumientos:

cadena **mysqli_character_set_name** (mysqli identificador_de_enlace)

Estilo orientado a objetos (método):

class **mysqli** {

cadena **character_set_name** (void)

}

Regresa el conjuntos de caracteres actual, determinado para la conexión de base de datos específicada por el parámetro *identificador_de_enlace*.

Valores retornados

El conjunto de caracteres por defecto, determinados para la conexión actual

Ver también

mysqli_client_encoding(), y **mysqli_real_escape_string()**.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
/* Open a connection */
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

/* Print current character set */
$charset = $mysqli->character_set_name();
printf ("Current character set is %s\n", $charset);

$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
/* Open a connection */
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (!$link) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

/* Print current character set */
$charset = mysqli_character_set_name($link);
printf ("Current character set is %s\n",$charset);

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Current character set is latin1_swedish_ci
```

mysqli_client_encoding

mysqli_client_encoding -- Alias de **mysqli_character_set_name()**

Descripción

Esta función es un alias de **mysqli_character_set_name()**. Para una descripción detallada vea **mysqli_character_set_name()**.

Ver también

mysqli_client_encoding(), y **mysqli_real_escape_string()**.

mysqli_close

(PHP 5)

mysqli_close

(no version information, might be only in CVS)

mysqli->close -- Cierra la conexión de base de datos previamente abierta

Descripción

Estilo por procedimientos:

bool **mysqli_close** (mysqli identificador_de_enlace)

Estilo orientado a objetos (método):

class **mysqli** {

bool **close** (void)

}

La función **mysqli_close()** cierra una conexión de base de datos previamente abierta, específicada por el parámetro *identificador_de_enlace*.

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Ver también

mysqli_connect(), **mysqli_init()**, y **mysqli_real_connect()**.

mysqli_commit

(PHP 5)

mysqli_commit

(no version information, might be only in CVS)

mysqli->commit -- Completa la transacción actual

Descripción

Estilo por procedimientos:

bool **mysqli_commit** (mysqli identificador_de_enlace)

Estilo orientado a objetos (método)

class **mysqli** {

bool **commit** (void)

}

Entrega la transacción actual para la conexión de base de datos específicada por el parametro *link*.

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Ver también

mysqli_autocommit(), y **mysqli_rollback()**.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$mysqli->query("CREATE TABLE Language LIKE CountryLanguage Type=InnoDB");

/* set autocommit to off */
$mysqli->autocommit(FALSE);

/* Insert some values */
$mysqli->query("INSERT INTO Language VALUES ('DEU', 'Bavarian', 'F', 11.2)");
$mysqli->query("INSERT INTO Language VALUES ('DEU', 'Swabian', 'F', 9.4)");

/* commit transaction */
$mysqli->commit();

/* drop table */
$mysqli->query("DROP TABLE Language");

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "test");

/* check connection */
if (!$link) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

/* set autocommit to off */
mysqli_autocommit($link, FALSE);

mysqli_query($link, "CREATE TABLE Language LIKE CountryLanguage Type=InnoDB");

/* Insert some values */
mysqli_query($link, "INSERT INTO Language VALUES ('DEU', 'Bavarian', 'F', 11.2)");
mysqli_query($link, "INSERT INTO Language VALUES ('DEU', 'Swabian', 'F', 9.4)");

/* commit transaction */
mysqli_commit($link);

/* close connection */
mysqli_close($link);
?>
```

mysqli_connect_errno

(PHP 5)

mysqli_connect_errno -- Regresa el código de error de la última llamada a la conexión

Descripción

int **mysqli_connect_errno** (void)

La función **mysqli_connect_errno()** regresará el últumo código de error generado por **mysqli_connect()**. Si no han ocurrido errores, esta función regresará cero.

> **Nota:** Los números de error del cliente, están listados en el archivo de MySQL `errmsg.h`. Los números de error del servidor, están listados en el archivo de MySQL `mysqld_error.h`. En la distribución de las fuentes de MySQL, tu puedes encontrar una lista completa de los mensajes de error y de los números de error en el documento `Docs/mysqld_error.txt`.

Valores retornados

Un código de error para la última llamada a **mysqli_connect()**, si ésta falla. Si no hay error entonces se regresa Cero.

Ver también

mysqli_connect(), **mysqli_connect_error()**, **mysqli_errno()**, **mysqli_error()**, y **mysqli_sqlstate()**

Ejemplos

Ejemplo 1. mysqli_connect_errno

```
<?php

$link = @mysqli_connect("localhost", "nonexisting_user", "");

if (!$link) {
    printf("Can't connect to localhost. Errorcode: %d\n", mysqli_connect_errno());
}
?>
```

mysqli_connect_error

(PHP 5)

mysqli_connect_error -- Regresa una descripción del último error de la conexión

Descripción

cadena **mysqli_connect_error** (void)

La función **mysqli_connect_error()** es identica a la función **mysqli_connect_errno()**, excepto en que en vez de regresar el número del código de error, la función **mysqli_connect_error()** regresará el mensaje de error correspondiente a la última llamada a la función **mysqli_connect()**. Si no ha ocurrido error, está función regresará una cadena vacía.

Valores retornados

Una cadena que describe el error o una cadena vacía si no ha ocurrido error.

Ver también

mysqli_connect(), **mysqli_connect_errno()**, **mysqli_errno()**, **mysqli_error()**, y **mysqli_sqlstate()**

Ejemplos

Ejemplo 1. mysqli_connect_error

```
<?php

$link = @mysqli_connect("localhost", "nonexisting_user", "");

if (!$link) {
    printf("Can't connect to localhost. Error: %s\n", mysqli_connect_error());
}
?>
```

mysqli_connect

(PHP 5)

mysqli_connect

(no version information, might be only in CVS)

mysqli() -- Abre una nueva conexión al servidor MySQL

Descripción

Estilo por procedimientos

mysqli **mysqli_connect** ([cadena equipo_anfitrión [, cadena usuario [, cadena contraseña [, cadena base_de_datos [, int puerto [, cadena socket]]]]]])

Estilo orientado a objetos (constructor):

class **mysqli** {

__construct ([cadena equipo_anfitrión [, cadena usuario [, cadena contraseña [, cadena base_de_datos [, int puerto [, cadena socket]]]]]])

}

La función **mysqli_connect()** intenta abrir una conexión al servidor MySQL que se está ejecutando en *equipo_afitrión* el cual puede ser el nombre de un equipo o una dirección IP. Pasando el valor **`NULL`** o la cadena "localhost" a este parámetro, se asume que está en el mismo equipo. Cuando sea posible se usarán "pipes" en vez del protocolo TCP/IP. En caso exitoso, la función **mysqli_connect ()** regresará un objeto representando la conexión a la base de datos, o **`FALSE`** en caso contrario.

En los parámetros *usuario* y *contraseña* se específica el nombre de usuario y contraseña con los cuales se debe conectar al servidor MySQL. Si no se da contraseña el valor **`NULL`** es tomado, el servidor MySQL intentará verificar al usuario contra los registros de usuarios que esten sin contraseña. Esto permite que un usuario pueda ser usado con diferentes permisos (dependiendo si se provee contraseña o no).

Si se específica el parámetro *base_de_datos* especificará la base de datos a usar por defecto cuando se ejecuten consultas.

Los parámetros *puerto* y *socket* son usados junto con el parámetro *equipo_anfitrión* para controla a futuro como conectar al servidor de base de datos. El parámetro *puerto* específica el número de puerto al que se intenta conectar en el servidor MySQL, mientras que el parámetro *socket* específica el socket o la pipa nombrada "pipe" que debe ser usada.

> **Nota:** Especificar el parámetro *socket* no determina explícitamente el tipo de conexión a ser usado cuando se conecta al servidor MySQL. El parámetro *equipo_anfitrión* determina como se hace la conexión a la base de datos MySQL.

Valores retornados

Regresa un objeto el cuál representa la conexión al servidor MySQL o **`FALSE`** en caso contrario.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

printf("Host information: %s\n", $mysqli->host_info);

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (!$link) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

printf("Host information: %s\n", mysqli_get_host_info($link));

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Host information: Localhost via UNIX socket
```

mysqli_data_seek

(PHP 5)

mysqli_data_seek

(no version information, might be only in CVS)

result->data_seek -- Ajusta el apuntador arbitrariamente a una fila en el resultado

Descripción

Estilo por procedimientos:

bool **mysqli_data_seek** (mysqli_result resultado, int posición)

Estilo orientado a objetos (método):

class **resultado** {

bool **data_seek** (int posición)

}

La función **mysqli_data_seek()** busca arbitrariamente apuntar al valor específicado por *posición* en

el conjunto resultante representado por *result*. Los valores del parámetro *posición* deben estar entre cero y el total de filas menus uno (0 .. **mysqli_num_rows()** - 1).

> **Nota:** Está función solo puede ser usada con resultados sin almacenamiento intermedio logrados por el uso de las funciones **mysqli_store_result()** o **mysqli_query()**.

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Ver también

mysqli_store_result(), **mysqli_fetch_row()**, **mysqli_fetch_array()**, **mysqli_fetch_assoc()**, **mysqli_fetch_object()**, **mysqli_query()**, y **mysqli_num_rows()**.

Ejemplos

Ejemplo 1. Estilo orientado a onjetos

```
<?php
/* Open a connection */
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER BY Name";
if ($result = $mysqli->query( $query)) {

    /* seek to row no. 400 */
    $result->data_seek(399);

    /* fetch row */
    $row = $result->fetch_row();

    printf ("City: %s  Countrycode: %s\n", $row[0], $row[1]);

    /* free result set*/
    $result->close();
}

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
/* Open a connection */
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (!$link) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER BY Name";

if ($result = mysqli_query($link, $query)) {

    /* seek to row no. 400 */
    mysqli_data_seek($result, 399);

    /* fetch row */
    $row = mysqli_fetch_row($result);

    printf ("City: %s  Countrycode: %s\n", $row[0], $row[1]);

    /* free result set*/
    mysqli_free_result($result);
}

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
City: Benin City  Countrycode: NGA
```

mysqli_debug

(PHP 5)

mysqli_debug -- Realiza operaciones de rastreo de errores

Descripción

void **mysqli_debug** (cadena acción)

La función **mysqli_debug()** es usada para realizar operaciones de rastreo y eliminación de errores utilizando la libreria "Fred Fish". El parámetro *acción* es una cadena que representa las operaciones de rastreo de errores a realizar.

> **Nota:** Para usar la funció **mysqli_debug()** se debe compilar el cliente de MySQL con soporte para rastreo de errores "debugging".

Valores retornados

mysqli_debug() no regresa ningún valor.

Ver también

mysqli_dump_debug_info(), y **mysqli_report()**

Ejemplos

Ejemplo 1. Generar un archivo de rastreo

```
<?php

/* Create a trace file in '/tmp/client.trace' on the local (client) machine: */
mysqli_debug("d:t:0,/tmp/client.trace");

?>
```

mysqli_disable_reads_from_master

(PHP 5)

mysqli_disable_reads_from_master

(no version information, might be only in CVS)

mysqli->disable_reads_from_master -- Deshabilita lectura del Servidor maestro

Descripción

Estilo por procedimientos:

void **mysqli_disable_reads_from_master** (mysqli link)

Estilo orientado a objetos (método):

class **mysqli** {

void **disable_reads_from_master** (void)

}

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

mysqli_disable_rpl_parse

(PHP 5)

mysqli_disable_rpl_parse -- Deshabilita el analizador RPL

Descripción

void **mysqli_disable_rpl_parse** (mysqli identificador_de_enlace)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

mysqli_dump_debug_info

(PHP 5)

mysqli_dump_debug_info

(no version information, might be only in CVS)

mysqli->dump_debug_info -- Vacía información de rastreo de errores en el log

Descripción

bool **mysqli_dump_debug_info** (mysqli identificador_de_enlace)

Esta función está diseñada para ser ejecutada por un usaurio con privilegios de SUPER usuario y es usada para vaciar la información de rastreo de errores en el log del servidor MySQL relacionado a la conexión específicada por el parámetro *identificador_de_enlace*.

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Ver también

mysqli_debug().

mysqli_embedded_connect

(PHP 5)

mysqli_embedded_connect -- Abre una conexió a un servidor MySQL embebido

Descripción

mysqli **mysqli_embedded_connect** ([string dbname])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

mysqli_enable_reads_from_master

(PHP 5)

mysqli_enable_reads_from_master -- Habilita la lectura del servidor Maestro

Descripción

void **mysqli_enable_reads_from_master** (mysqli identificador_de_enlace)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

mysqli_enable_rpl_parse

(PHP 5)

mysqli_enable_rpl_parse -- Habilita el analizador RPL

Descripción

void **mysqli_enable_rpl_parse** (mysqli identificador_de_enlace)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

mysqli_errno

(PHP 5)

mysqli_errno

(no version information, might be only in CVS)

mysqli->errno -- Regresa el código de error para la función más recientemente llamada

Descripción

Estilo por procedimientos:

int **mysqli_errno** (mysqli identificador_de_enlace)

Estilo orientado a objetos (propiedad):

class **mysqli** {

int errno

}

La función **mysqli_errno()** regresará el último código de error para la función de MySQLi más recientemente llamada, que pueda ser exitosa o fallar con respecto al identificador de enlace a la base de datos definido por el parámetro *identificador_de_enlace*. Si no han ocurrido errores, está función regresará cero.

> **Nota:** Los números de error del cliente, están listados en el archivo de MySQL `errmsg.h`. Los números de error del servidor, están listados en el archivo de MySQL `mysqld_error.h`. En la distribución de las fuentes de MySQL, tu puedes encontrar una lista completa de los mensajes de error y de los números de error en el documento `Docs/mysqld_error.txt`.

Valores retornados

Un valor de código de error para la última llamada si falló. Cero significa que no han ocurrido errores.

Ver también

mysqli_connect_errno(), **mysqli_connect_error()**, **mysqli_error()**, y **mysqli_sqlstate()**

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

if (!$mysqli->query("SET a=1")) {
    printf("Errorcode: %d\n", $mysqli->errno);
}

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

if (!mysqli_query($link, "SET a=1")) {
    printf("Errorcode: %d\n", mysqli_errno($link));
}

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Errorcode: 1193
```

mysqli_error

(PHP 5)

mysqli_error -- Regresa una cadena con la descripció del último error

Descripción

Estilo por procedimientos:

cadena **mysqli_error** (mysqli identificador_de_enlace)

Estilo orientado a objetos (propiedad)

class **mysqli** {

cadena error

}

La funció **mysqli_error()** es identica a la correspondiente **mysqli_errno()** en todos los sentidos, excepto en que en lugar de regresar un valor numérico, la función **mysqli_error()** regresará un mensaje de error, representando el último error ocurrido para la conexió de base de datos *identificador_de_enlace*. Si no han ocurrido errores, está función regresará una cadena vacía.

Valores retornados

Una cadena que describe el error. Una cadena vacía si no han ocurrido errores.

Ver también

mysqli_connect_errno(), **mysqli_connect_error()**, **mysqli_errno()**, y **mysqli_sqlstate()**

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

if (!$mysqli->query("SET a=1")) {
    printf("Errormessage: %s\n", $mysqli->error);
}

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

if (!mysqli_query($link, "SET a=1")) {
    printf("Errormessage: %s\n", mysqli_error($link));
}

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Errormessage: Unknown system variable 'a'
```

mysqli_escape_string

mysqli_escape_string -- Alias de **mysqli_real_escape_string()**

Descripción

Esta función es un alias de **mysqli_real_escape_string()**.

mysqli_execute

mysqli_execute -- Alias de **mysqli_stmt_execute()**

Descripción

Esta función es un alias de **mysqli_stmt_execute()**. Para una descipció más detallada vea la descripción de **mysqli_stmt_execute()**.

> **Nota: mysqli_execute()** es obsoleta y será removida.

Ver también

mysqli_stmt_execute().

mysqli_fetch_array

(PHP 5)

mysqli_fetch_array

(no version information, might be only in CVS)

result->fetch_array -- Obtiene una fila como una matriz asociativa, una matriz numérica o ambos

Descripción

Estilo por procedimientos:

mixto **mysqli_fetch_array** (mysqli resultado [, int tipo_de_resultado])

Estilo orientado a objetos (método):

class **mysqli_resultado** {

mixto **fetch_array** ([int tipo_de_resultado])

}

Regresa una matrix que corresponde a las filas obtenidas o **`NULL`** si no hay más filas para el resultado, representado por el parámetro *resultado*.

mysqli_fetch_array() es una versión mejorada de la función **mysqli_fetch_row()**. Ademá de almacenar los datos en índices numéricos de la matriz resultante, la función **mysqli_fetch_array()** también puede almacenar los datos en índices asociativos, usando los nombre de los campos de el resultado como llaves.

> **Nota:** Los nombres de campos retornados por esta función diferencian entre *mayusculas y minusculas*.

Si dos o más columnas de el resultado tienen el mismo nombre, la última columna tomara precedencia y sobre escribirá lo primero. Para acceder a varias columnas con el mismo nombre, la forma de índice numérica debe ser usada.

El parámetro opcional *tipo_de_resultado* es una constante que indica qué tipo de matriz debe ser producido para la fila de datos actual. Los posibles valires para este parámetro son las constantes MYSQLI_ASSOC, MYSQLI_NUM, o MYSQLI_BOTH. Por defecto la función **mysqli_fetch_array()** asumirá el valor de MYSQLI_BOTH.

Al usar la constante MYSQLI_ASSOC, esta función se comportará identica a la función **mysqli_fetch_assoc()**, mientras que con MYSQLI_NUM se comportará identica a la función

mysqli_fetch_row(). La opción final MYSQLI_BOTH creará una matriz con los atributos de ambos.

Valores retornados

Regresa una matriz que corresponde a las filas obtenidas o **`NULL`** si no hay más filas en el resultado.

Ver también

mysqli_fetch_assoc(), **mysqli_fetch_row()**, **mysqli_fetch_object()**, **mysqli_query()**, y **mysqli_data_seek()**.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER by ID LIMIT 3";
$result = $mysqli->query($query);

/* numeric array */
$row = $result->fetch_array(MYSQLI_NUM);
printf ("%s (%s)\n", $row[0], $row[1]);

/* associative array */
$row = $result->fetch_array(MYSQLI_ASSOC);
printf ("%s (%s)\n", $row["Name"], $row["CountryCode"]);

/* associative and numeric array */
$row = $result->fetch_array(MYSQLI_BOTH);
printf ("%s (%s)\n", $row[0], $row["CountryCode"]);

/* free result set */
$result->close();

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER by ID LIMIT 3";
$result = mysqli_query($link, $query);

/* numeric array */
$row = mysqli_fetch_array($result, MYSQLI_NUM);
printf ("%s (%s)\n", $row[0], $row[1]);

/* associative array */
$row = mysqli_fetch_array($result, MYSQLI_ASSOC);
printf ("%s (%s)\n", $row["Name"], $row["CountryCode"]);

/* associative and numeric array */
$row = mysqli_fetch_array($result, MYSQLI_BOTH);
printf ("%s (%s)\n", $row[0], $row["CountryCode"]);

/* free result set */
mysqli_free_result($result);

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Kabul (AFG)
Qandahar (AFG)
Herat (AFG)
```

mysqli_fetch_assoc

(PHP 5)

mysqli_fetch_assoc

(no version information, might be only in CVS)

mysqli->fetch_assoc -- Obtiene una fila del resultado como una matriz asociativa

Descripción

Estilo por procedimientos:

matriz **mysqli_fetch_assoc** (mysqli_result resultado)

Estilo orientado a objetos (método):

class **resultado** {

matriz **fetch_assoc** (void)

}

Regresa una matriz asociativa que corresponde a las filas obtenidas o **NULL** si no hay mas filas.

La función **mysqli_fetch_assoc()** es usada para regresar una representación asociativa de la siguiente fila en el resultado, representado por el parámetro *resultado*, donde cada llave en la matriz representa el nombre de las columnas en el resultado.

Si dos o más columnas de el resultado tienen el mismo nombre, la última columna tomara precedencia y sobre escribirá lo primero. Para acceder a varias columnas con el mismo nombre, la forma de índice numérica debe ser usada.

> **Nota:** Los nombres de campos retornados por esta función diferencian entre *mayusculas y minusculas*.
>
> **Nota:** Esta funcion define campos NULL como valores PHP **NULL**.

Valores retornados

Regresa una matriz que corresponde a las filas obtenidas o **NULL** si no hay más filas en el resultado.

Ver también

mysqli_fetch_array(), **mysqli_fetch_row()**, **mysqli_fetch_object()**, **mysqli_query()**, y **mysqli_data_seek()**.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER by ID DESC LIMIT 50,5";

if ($result = $mysqli->query($query)) {

    /* fetch associative array */
    while ($row = $result->fetch_assoc()) {
        printf ("%s (%s)\n", $row["Name"], $row["CountryCode"]);
    }

    /* free result set */
    $result->close();
}

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER by ID DESC LIMIT 50,5";

if ($result = mysqli_query($link, $query)) {

    /* fetch associative array */
    while ($row = mysqli_fetch_assoc($result)) {
        printf ("%s (%s)\n", $row["Name"], $row["CountryCode"]);
    }

    /* free result set */
    mysqli_free_result($result);
}

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Pueblo (USA)
Arvada (USA)
Cape Coral (USA)
Green Bay (USA)
Santa Clara (USA)
```

mysqli_fetch_field_direct

(PHP 5)

mysqli_fetch_field_direct

(no version information, might be only in CVS)

result->fetch_field_direct -- Obtiene los metadatos de un campo

Descripción

Estilo por procedimientos:

mixto **mysqli_fetch_field_direct** (mysqli_result resultado, int indice_de_campo)

Estilo orientado a objetos (método):

class **mysqli_result** {

mixto **fetch_field_direct** (int indice_de_campo)

}

La función **mysqli_fetch_field_direct()** regresa un objeto el cuál contiene información de la definición del campo del resultado específicado. El valor de indice_de_campo debe estar en el rango

de *0* a *número de campos -1*.

Valores retornados

Regresa un objeto el cual contiene información de la definición del campo o **`FALSE`** si no hay información para el *indice_de_campo* específicado.

Tabla 1. Atributos del objeto

Atributo	Descripción
name	Nombre de la columna
orgname	Nombre original de la columna si se dió un alias
table	Nombre de la tabla a la que pertenece el campo
orgtable	Nombre original de la tabla si se dió un alias
def	El valor por defecto para este campo, representado como una cadena
max_length	La amplitud máxima de campo de el campo para el resultado
flags	Un entero que representa los bit bandera para el campo
type	Tipo de dato utilizado para este campo
decimals	Número de decimales usadas (para campos entero)

Ver también

mysqli_num_fields(), **mysqli_fetch_field()**, y **mysqli_fetch_fields()**.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```php
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, SurfaceArea from Country ORDER BY Name LIMIT 5";

if ($result = $mysqli->query($query)) {

    /* Get field information for column 'SurfaceArea' */
    $finfo = $result->fetch_field_direct(1);

    printf("Name:     %s\n", $finfo->name);
    printf("Table:    %s\n", $finfo->table);
    printf("max. Len: %d\n", $finfo->max_length);
    printf("Flags:    %d\n", $finfo->flags);
    printf("Type:     %d\n", $finfo->type);

    $result->close();
}

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```php
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, SurfaceArea from Country ORDER BY Name LIMIT 5";

if ($result = mysqli_query($link, $query)) {

    /* Get field information for column 'SurfaceArea' */
    $finfo = mysqli_fetch_field_direct($result, 1);

    printf("Name:     %s\n", $finfo->name);
    printf("Table:    %s\n", $finfo->table);
    printf("max. Len: %d\n", $finfo->max_length);
    printf("Flags:    %d\n", $finfo->flags);
    printf("Type:     %d\n", $finfo->type);

    mysqli_free_result($result);
}

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Name:     SurfaceArea
Table:    Country
max. Len: 10
Flags:    32769
Type:     4
```

mysqli_fetch_field

(PHP 5)

mysqli_fetch_field

(no version information, might be only in CVS)

result->fetch_field -- Regresa metadatos de el campo en el resultado

Descripción

Estilo por procedimientos:

mixto **mysqli_fetch_field** (mysqlI_result resultado)

Estilo orientado a objetos (método):

class **mysqli_result** {

mixto **fetch_field** (void)

}

La función **mysqli_fetch_field()** regresa un objeto con información de definición de un campo de un resultado. Ejecute esta función repetidamente para obtener la información de todas las columnas en un resultado. **mysqli_fetch_field()** regresa **`FALSE`** cuando no hay más columnas en el resultado.

Valores retornados

Regresa un objeto el cual contiene la información de la definición del campo o **`FALSE`** si no hay información disponible.

Tabla 1. Propiedades del objeto

Propiedad	Descripción
name	El nombre de la columna
orgname	Nombre original de la columna, si se específico un alias
table	El nombre de la tablaa la cuál pertenece
orgtable	Nombre original de la tabla si se específico un alias
def	El valor por defecto de este campo, representado con una cadena
max_length	La máma amplitud del campo para el resultado
flags	Un entero representando los bit de bandera para el campo
type	El tipo de dato usado para este campo
decimals	El número de decimales utilizadas (para campos numéricos)

Ver también

mysqli_num_fields(), **mysqli_fetch_field_direct()**, **mysqli_fetch_fields()**, y **mysqli_field_seek()**

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, SurfaceArea from Country ORDER BY Code LIMIT 5";

if ($result = $mysqli->query($query)) {

    /* Get field information for all columns */
    while ($finfo = $result->fetch_field()) {

        printf("Name:     %s\n", $finfo->name);
        printf("Table:    %s\n", $finfo->table);
        printf("max. Len: %d\n", $finfo->max_length);
        printf("Flags:    %d\n", $finfo->flags);
        printf("Type:     %d\n\n", $finfo->type);
    }
    $result->close();
}

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, SurfaceArea from Country ORDER BY Code LIMIT 5";

if ($result = mysqli_query($link, $query)) {

    /* Get field information for all fields */
    while ($finfo = mysqli_fetch_field($result)) {

        printf("Name:     %s\n", $finfo->name);
        printf("Table:    %s\n", $finfo->table);
        printf("max. Len: %d\n", $finfo->max_length);
        printf("Flags:    %d\n", $finfo->flags);
        printf("Type:     %d\n\n", $finfo->type);
    }
    mysqli_free_result($result);
}

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Name:     Name
Table:    Country
max. Len: 11
Flags:    1
Type:     254

Name:     SurfaceArea
Table:    Country
max. Len: 10
Flags:    32769
Type:     4
```

mysqli_fetch_fields

(PHP 5)

mysqli_fetch_fields

(no version information, might be only in CVS)

result->fetch_fields -- Regresa una matriz de objetos representando los campos en un resultado

Descripción

Estilo por procedimientos:

mixto **mysqli_fetch_fields** (mysqli_result resultado)

Estilo orientado a objetos (método):

class **mysqli_result** {

mixto **fetch_fields** (void)

}

Esta función sirve identica en propósito a la función **mysqli-fetch-field()** con la minima diferencia de que, en lugar de regresar un objeto a la vez para cada campo, las columnas son regresadas como una matiz de objetos.

Valores retornados

Regresa una matriz de objetos que contienen la información de definició de los campos o **FALSE** si no hay información del campo disponible.

Tabla 1. Propiedades del objeto

Propiedad	Descripción
name	El nombre de la columna
orgname	Nombre original de la columna, si se específico un alias
table	El nombre de la tablaa la cuál pertenece
orgtable	Nombre original de la tabla si se específico un alias

Propiedad	Descripción
def	El valor por defecto de este campo, representado con una cadena
max_length	La máma amplitud del campo para el resultado
flags	Un entero representando los bit de bandera para el campo
type	El tipo de dato usado para este campo
decimals	El número de decimales utilizadas (para campos numéricos)

Ver también

mysqli_num_fields(), **mysqli_fetch_field()**, y **mysqli_fetch_field_direct()**.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, SurfaceArea from Country ORDER BY Code LIMIT 5";

if ($result = $mysqli->query($query)) {

    /* Get field information for all columns */
    $finfo = $result->fetch_fields();

    for ($i=0; $i < count($finfo); $i++) {
        printf("Name:     %s\n", $finfo[$i]->name);
        printf("Table:    %s\n", $finfo[$i]->table);
        printf("max. Len: %d\n", $finfo[$i]->max_length);
        printf("Flags:    %d\n", $finfo[$i]->flags);
        printf("Type:     %d\n\n", $finfo[$i]->type);
    }
    $result->close();
}

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, SurfaceArea from Country ORDER BY Code LIMIT 5";

if ($result = mysqli_query($link, $query)) {

    /* Get field information for all columns */
    $finfo = mysqli_fetch_fields($result);

    for ($i=0; $i < count($finfo); $i++) {
        printf("Name:     %s\n", $finfo[$i]->name);
        printf("Table:    %s\n", $finfo[$i]->table);
        printf("max. Len: %d\n", $finfo[$i]->max_length);
        printf("Flags:    %d\n", $finfo[$i]->flags);
        printf("Type:     %d\n\n", $finfo[$i]->type);
    }
    mysqli_free_result($result);
}

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Name:     Name
Table:    Country
max. Len: 11
Flags:    1
Type:     254

Name:     SurfaceArea
Table:    Country
max. Len: 10
Flags:    32769
Type:     4
```

mysqli_fetch_lengths

(PHP 5)

mysqli_fetch_lengths

(no version information, might be only in CVS)

result->lengths -- Regresa la longitud de las columnas de la fila actual en el resultado

Descripción

Estilo por procedimientos:

mixto **mysqli_fetch_lengths** (mysqli_result resultado)

Estilo orientado a objetos(propiedad):

class **mysqli_result** {

mixto lengths

}

La función **mysqli_fetch_lengths()** regresa una matriz conteniendo la longitud de cada columna de la fila actual en el resultado representado por el parámetro *result*. Si hay información regresa una matriz numéricamente ordenada representando la longitud de cada columna o **FALSE** si falla.

Valores retornados

Una matriz de enteros representando el tamaño de cada columna (sin incluir el caracter NULL al final de la columna). **FALSE** si ocurre un error.

mysqli_fetch_lengths() Es valida solo para la fila actual en el resultado. Regresa **FALSE** si se llama antes de ejecutar mysqli_fetch_row/array/object o después de obtener todas las filas en el resultado.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$query = "SELECT * from Country ORDER BY Code LIMIT 1";

if ($result = $mysqli->query($query)) {

    $row = $result->fetch_row();

    /* display column lengths */
    for ($i=0; $i < count($result->lengths); $i++) {
        printf("Field %2d has Length %2d\n", $i+1, $result->lengths[$i]);
    }
    $result->close();
}

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$query = "SELECT * from Country ORDER BY Code LIMIT 1";

if ($result = mysqli_query($link, $query)) {

          $row = mysqli_fetch_row($result);

    /* display column lengths */
    $lengths = mysqli_fetch_lengths($result);
    for ($i=0; $i < count($lengths); $i++) {
        printf("Field %2d has Length %2d\n", $i+1, $lengths[$i]);
    }
    mysqli_free_result($result);
}

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Field  1 has Length  3
Field  2 has Length  5
Field  3 has Length 13
Field  4 has Length  9
Field  5 has Length  6
Field  6 has Length  1
Field  7 has Length  6
Field  8 has Length  4
Field  9 has Length  6
Field 10 has Length  6
Field 11 has Length  5
Field 12 has Length 44
Field 13 has Length  7
Field 14 has Length  3
Field 15 has Length  2
```

mysqli_fetch_object

(PHP 5)

mysqli_fetch_object

(no version information, might be only in CVS)

result->fetch_object -- Regresa la fila actual del resultado como un objeto

Descripción

Estilo por procedimientos:

mixto **mysqli_fetch_object** (mysqli_result resultado)

Estilo orientado a objetos (método):

class **mysqli_result** {

mixto **fetch_object** (void)

}

La función **mysqli_fetch_object()** regresará la fila actual en el resultado como un objeto donde los atributos del objeto representan los nombre de los campos encontrados dentro del resultado. Si no hay más filas en el resultado, se regresa **NULL**

Valores retornados

Regresa un objeto que corresponde a la fila obtenida o **NULL** si no hay más filas en el resultado.

> **Nota:** Los nombres de campos retornados por esta función diferencian entre *mayusculas y minusculas*.
>
> **Nota:** Esta funcion define campos NULL como valores PHP **NULL**.

Ver también

mysqli_fetch_array(), **mysqli_fetch_assoc()**, **mysqli_fetch_row()**, **mysqli_query()**, y **mysqli_data_seek()**.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER by ID DESC LIMIT 50,5";

if ($result = $mysqli->query($query)) {

    /* fetch object array */
    while ($obj = $result->fetch_object()) {
        printf ("%s (%s)\n", $obj->Name, $obj->CountryCode);
    }

    /* free result set */
    $result->close();
}

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER by ID DESC LIMIT 50,5";

if ($result = mysqli_query($link, $query)) {

    /* fetch associative array */
    while ($obj = mysqli_fetch_object($result)) {
        printf ("%s (%s)\n", $obj->Name, $obj->CountryCode);
    }

    /* free result set */
    mysqli_free_result($result);
}

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Pueblo (USA)
Arvada (USA)
Cape Coral (USA)
Green Bay (USA)
Santa Clara (USA)
```

mysqli_fetch_row

(PHP 5)

mysqli_fetch_row

(no version information, might be only in CVS)

result->fetch_row -- Obtiene una fila del resultado como una matriz enumerada

Descripción

Estilo por procedimientos:

mixto **mysqli_fetch_row** (mysqli_result resultado)

Estilo orientado a objetos (método):

class **mysqli_result** {

mixto **fetch_row** (void)

}

Regresa una matriz que corresponde a la fila obtenida, o **NULL** si no hay más filas.

La función **mysqli_fetch_row()** obtiene una fila de datos del resultado, representado por el parámetro *result* y lo regresa como una matriz enumerada, donde cada columna es almacenada en la matriz empezando de 0 (cero). Cada llamada subsecuente a la función **mysqli_fetch_row()** regresará la siguiente fila dentro del resultado, o **NULL** si no hay más filas.

Valores retornados

mysqli_fetch_row() regresa una matriz que corresponde a la fila obtenida o **NULL** si no hay más filas en el resultado.

> **Nota:** Los nombres de campos retornados por esta función diferencian entre *mayusculas y minusculas*.

> **Nota:** Esta funcion define campos NULL como valores PHP **NULL**.

Ver también

mysqli_fetch_array(), **mysqli_fetch_assoc()**, **mysqli_fetch_object()**, **mysqli_query()**, y **mysqli_data_seek()**.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER by ID DESC LIMIT 50,5";

if ($result = $mysqli->query($query)) {

    /* fetch object array */
    while ($row = $result->fetch_row()) {
        printf ("%s (%s)\n", $row[0], $row[1]);
    }

    /* free result set */
    $result->close();
}

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER by ID DESC LIMIT 50,5";

if ($result = mysqli_query($link, $query)) {

    /* fetch associative array */
    while ($row = mysqli_fetch_row($result)) {
        printf ("%s (%s)\n", $row[0], $row[1]);
    }

    /* free result set */
    mysqli_free_result($result);
}

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Pueblo (USA)
Arvada (USA)
Cape Coral (USA)
Green Bay (USA)
Santa Clara (USA)
```

mysqli_fetch

mysqli_fetch -- Alias de **mysqli_stmt_fetch()**

Descripción

Esta función es un alias de **mysqli_stmt_fetch()**. Para una descripción más detallada vea **mysqli_stmt_fetch()**.

> **Nota: mysqli_fetch()** es obsoleta y será removida.

Ver también

mysqli_stmt_fetch().

mysqli_field_count

(PHP 5)

mysqli_field_count

(no version information, might be only in CVS)

mysqli->field_count -- Regresa el número de columnas para la consulta más reciente

Descripción

Estilo por procedimientos:

int **mysqli_field_count** (mysqli identificador_de_enlace)

Estilo orientado a objetos (método):

class **mysqli** {

int **field_count** (void)

}

Regresa el número de columnas para la más reciente consulta en la conexión representada por el parámetro *identificador_de_enlace*. Esta función puede ser usada cuando se use la función Returns the number of columns for the most recent query on the connection represented by the *link* parameter. This function can be useful when using the **mysqli_store_result()** function to determine if the query should have produced a non-empty result set or not without knowing the nature of the query.

Valores retornados

An integer representing the number of fields in a result set

Ejemplos

Ejemplo 1. Object oriented style

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "test");

$mysqli->query( "DROP TABLE IF EXISTS friends");
$mysqli->query( "CREATE TABLE friends (id int, name varchar(20))");

$mysqli->query( "INSERT INTO friends VALUES (1,'Hartmut'), (2, 'Ulf')");


$mysqli->real_query($HTTP_POST_VARS['query']);

if (mysqli_field_count($link)) {
    /* this was a select/show or describe query */
    $result = $mysqli->store_result();

    /* process resultset */
    $row = $result->fetch_row();

    /* free resultset */
    $result->close();
}

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Procedural style

```php
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "test");

mysqli_query($link, "DROP TABLE IF EXISTS friends");
mysqli_query($link, "CREATE TABLE friends (id int, name varchar(20))");

mysqli_query($link, "INSERT INTO friends VALUES (1,'Hartmut'), (2, 'Ulf')");

mysqli_real_query($link, $HTTP_POST_VARS['query']);

if (mysqli_field_count($link)) {
    /* this was a select/show or describe query */
    $result = mysqli_store_result($link);

    /* process resultset */
    $row = mysqli_fetch_row($result);

    /* free resultset */
    mysqli_free_result($result);
}

/* close connection */
mysqli_close($link);
?>
```

mysqli_field_seek

(PHP 5)

mysqli_field_seek

(no version information, might be only in CVS)

result->field_seek -- Fija el apuntador del resultado a una posicion específica

Descripción

Estilo por procedimientos:

int **mysqli_field_seek** (mysqli_result resultado, int fieldnr)

Estilo orientado a objetos (método):

class **mysqli_result** {

int **field_seek** (int fieldnr)

}

Fija el apuntador a la posición dada. En la siguiente llamada a la función **mysqli_fetch_field()**, traerá la definición de la columna asociada con esa posición.

> **Nota:** Para encontrar el inicio de una fila, pase una posición con valor cero.

Valores retornados

mysqli_field_seek() Regresa la posición previa del apuntador de campo.

Ver también

mysqli_fetch_field().

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, SurfaceArea from Country ORDER BY Code LIMIT 5";

if ($result = $mysqli->query($query)) {

    /* Get field information for 2nd column */
    $result->field_seek(1);
    $finfo = $result->fetch_field();

    printf("Name:     %s\n", $finfo->name);
    printf("Table:    %s\n", $finfo->table);
    printf("max. Len: %d\n", $finfo->max_length);
    printf("Flags:    %d\n", $finfo->flags);
    printf("Type:     %d\n\n", $finfo->type);

    $result->close();
}

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, SurfaceArea from Country ORDER BY Code LIMIT 5";

if ($result = mysqli_query($link, $query)) {

    /* Get field information for 2nd column */
    mysqli_field_seek($result, 1);
    $finfo = mysqli_fetch_field($result);

    printf("Name:     %s\n", $finfo->name);
    printf("Table:    %s\n", $finfo->table);
    printf("max. Len: %d\n", $finfo->max_length);
    printf("Flags:    %d\n", $finfo->flags);
    printf("Type:     %d\n\n", $finfo->type);

    mysqli_free_result($result);
}

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Name:     SurfaceArea
Table:    Country
max. Len: 10
Flags:    32769
Type:     4
```

mysqli_field_tell

(PHP 5)

mysqli_field_tell

(no version information, might be only in CVS)

result->current_field -- Obtiene la posición de desplazamiento del apuntador de un resultado

Descripción

Estilo por procedimientos:

int **mysqli_field_tell** (mysqli_result resultado)

Estilo orientado a objetos (propiedad):

class **mysqli_result** {

int current_field

}

Regresa la posición del apuntador usada en la última llamada a **mysqli_fetch_field()**. Este valor puede ser usada como un argumento de **mysqli_field_seek()**.

Valores retornados

Regresa la posición actual del apuntador de campo.

Ver también

mysqli_fetch_field(), y **mysqli_field_seek()**.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```php
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, SurfaceArea from Country ORDER BY Code LIMIT 5";

if ($result = $mysqli->query($query)) {

    /* Get field information for all columns */
    while ($finfo = $result->fetch_field()) {

        /* get fieldpointer offset */
        $currentfield = $result->current_field;

        printf("Column %d:\n", $currentfield);
        printf("Name:     %s\n", $finfo->name);
        printf("Table:    %s\n", $finfo->table);
        printf("max. Len: %d\n", $finfo->max_length);
        printf("Flags:    %d\n", $finfo->flags);
        printf("Type:     %d\n\n", $finfo->type);
    }
    $result->close();
}

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```php
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, SurfaceArea from Country ORDER BY Code LIMIT 5";

if ($result = mysqli_query($link, $query)) {

    /* Get field information for all fields */
    while ($finfo = mysqli_fetch_field($result)) {

        /* get fieldpointer offset */
        $currentfield = mysqli_field_tell($result);

        printf("Column %d:\n", $currentfield);
        printf("Name:     %s\n", $finfo->name);
        printf("Table:    %s\n", $finfo->table);
        printf("max. Len: %d\n", $finfo->max_length);
        printf("Flags:    %d\n", $finfo->flags);
        printf("Type:     %d\n\n", $finfo->type);
    }
    mysqli_free_result($result);
}

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Column 1:
Name:     Name
Table:    Country
max. Len: 11
Flags:    1
Type:     254

Column 2:
Name:     SurfaceArea
Table:    Country
max. Len: 10
Flags:    32769
Type:     4
```

mysqli_free_result

(PHP 5)

mysqli_free_result

(no version information, might be only in CVS)

result->free -- Libera la memoria asociada con el resultado

Descripción

Estilo por procedimientos:

void **mysqli_free_result** (mysqli_result resultado)

Estilo orientado a objetos (método):

class **mysqli_result** {

void **free** (void)

}

La función **mysqli_free_result()** libera la memoria asociada al resultado representado por el parámetro *resultado*, la cuál fue tomada por **mysqli_query()**, **mysqli_store_result()** o **mysqli_use_result()**.

> **Nota:** Usted debe siempre liberar el resultado con **mysqli_free_result()**, cuando su objeto resultado no se necesite más.

Valores retornados

Esta función no regresa ningun valor.

Ver también

mysqli_query(), **mysqli_stmt_store_result()**, **mysqli_store_result()**, y **mysqli_use_result()**.

mysqli_get_client_info

(PHP 5)

mysqli_get_client_info -- Regresa en una cadena la versión del cliente de MySQL

Descripción

cadena **mysqli_get_client_info** (void)

La funció **mysqli_get_client_info()** es usada para regresar una cadena representando la versión del cliente utilizada en la extensión MySQLi.

Valores retornados

Una cadena que representa la versión de la libreria cliente de MySQL

Ver también

mysqli_get_client_version(), **mysqli_get_server_info()**, y **mysqli_get_server_version()**.

Ejemplos

Ejemplo 1. mysqli_get_client_info

```
<?php

/* We don't need a connection to determine
   the version of mysql client library */

printf("Client library version: %s\n", mysqli_get_client_info());
?>
```

mysqli_get_client_version

(PHP 5)

mysqli_get_client_version -- Obtiene información del cliente MySQL

Descripción

int **mysqli_get_client_version** (void)

Regresa el número de la versión del cliente como un entero.

Valores retornados

Un número que representa la versión de la libreria del cliente de MySQL en el formato : *main_version*10000 + minor_version *100 + sub_version*. Por ejemplo, 4.1.0 es regresado como 40100.

Esto es útil para determinar rápidamente la version de las librerias del cliente para saber si existen algunas capacidades.

Ver también

mysqli_get_client_info(), **mysqli_get_server_info()**, y **mysqli_get_server_version()**.

Ejemplos

Ejemplo 1. mysqli_get_client_version

```
<?php

/* We don't need a connection to determine
   the version of mysql client library */

printf("Client library version: %d\n", mysqli_get_client_version());
?>
```

mysqli_get_host_info

(PHP 5)

mysqli_get_host_info

(no version information, might be only in CVS)

mysqli->get_host_info -- Regresa una cadena que representa el tipo de conexión usada

Descripción

Estilo por procedimientos:

cadena **mysqli_get_host_info** (mysqli identificador_de_enlace)

Estilo orientado a objetos (propiedad):

class **mysqli** {

cadena host_info

}

La función **mysqli_get_host_info()** regresa una cadena describiendo la conexión utilizada, representada por el parámetro *identificador_de_enlace* (incluyendo el nombre del servidor).

Valores retornados

Una cadena representando el nombre del servidor y tipo de conexión.

Ver también

mysqli_get_proto_info().

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

/* print host information */
printf("Host info: %s\n", $mysqli->host_info);

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

/* print host information */
printf("Host info: %s\n", mysqli_get_host_info($link));

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Host info: Localhost via UNIX socket
```

mysqli_get_metadata

mysqli_get_metadata -- Alias para **mysqli_stmt_result_metadata()**

Descripción

Esta función es un alias de **mysqli_stmt_result_metadata()**. Para una descripción detallada vea **mysqli_stmt_result_metadata()**.

> **Nota: mysqli_get_metadata()** es obsoleta y será removida.

Ver también

mysqli_stmt_result_metadata().

mysqli_get_proto_info

(PHP 5)

mysqli_get_proto_info

(no version information, might be only in CVS)

mysqli->protocol_version -- Regresa la versión del protocolo de MySQL usado

Descripción

Estilo por procedimientos:

int **mysqli_get_proto_info** (mysqlito identificador_de_enlace)

Estilo orientado a objetos (propiedad):

class **mysqli** {

cadena protocol_version

}

Regresa un entero que representa la verión del protocolo MySQL usado por la conexión representada por el parámetro *identificador_de_enlace*.

Valores retornados

Regresa un entero representando la versión del protocolo.

Ver también

mysqli_get_host_info().

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

/* print protocol version */
printf("Protocol version: %d\n", $mysqli->protocol_version);

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

/* print protocol version */
printf("Protocol version: %d\n", mysqli_get_proto_info($link));

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Protocol version: 10
```

mysqli_get_server_info

(PHP 5)

mysqli_get_server_info

(no version information, might be only in CVS)

mysqli->server_info -- Regresa la version del servidor de MySQL

Descripción

Estilo por procedimientos:

cadena **mysqli_get_server_info** (mysqli identificador_de_enlace)

Estilo orientado a objetos (propiedad):

class **mysqli** {

cadena server_info

}

Regresa una cadena representando la versión del servidor MySQL a la que la extensión de MySQLi está conectado (representado por el parámetro *identificador_de_enlace*).

Valores retornados

Una cadena representando la versión del servidor.

Ver también

mysqli_get_client_info(), **mysqli_get_client_version()**, y **mysqli_get_server_version()**.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

/* print server version */
printf("Server version: %s\n", $mysqli->server_info);

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

/* print server version */
printf("Server version: %s\n", mysqli_get_server_info($link));

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Server version: 4.1.2-alpha-debug
```

mysqli_get_server_version

(PHP 5)

mysqli_get_server_version -- Regresa la versión del servidor MySQL como un entero

Descripción

Estilo por procedimientos:

int **mysqli_get_server_version** (mysqli identificador_de_enlace)

Estilo orientado a objetos (propiedad):

class **mysqli** {

int server_version

}

La función **mysqli_get_server_version()** regresa la versión del servidor al que está conectado (representado por el parámetro *identificador_de_enlace*) como un entero.

La forma del número de version es *main_version * 10000 + minor_version * 100 + sub_version* (i.e. version 4.1.0 is 40100).

Valores retornados

Un entero representando la versión del servidor.

Ver también

mysqli_get_client_info(), **mysqli_get_client_version()**, y **mysqli_get_server_info()**.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

/* print server version */
printf("Server version: %d\n", $mysqli->server_version);

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

/* print server version */
printf("Server version: %d\n", mysqli_get_server_version($link));

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Server version: 40102
```

mysqli_info

(PHP 5)

mysqli_info

(no version information, might be only in CVS)

mysqli->info -- Obtiene información acerca de la consulta más recientemente ejecutada

Descripción

Estilo por procedimientos:

cadena **mysqli_info** (mysqli identificador_de_enlace)

Estilo orientado a objetos (propiedad)

class **mysqli** {

cadena info

}

La función **mysqli_info()** regresa una cadena proveyendo información acerca de la consulta ejecutada más recientemente. La naturaleza de esta cadena es como sigue:

Tabla 1. Valores regresados posibles de mysqli_info

Tipo de Consulta	Ejemplo de cadena resultante
INSERT INTO...SELECT...	Records: 100 Duplicates: 0 Warnings: 0
INSERT INTO...VALUES (...),(...),(...)	Records: 3 Duplicates: 0 Warnings: 0
LOAD DATA INFILE ...	Records: 1 Deleted: 0 Skipped: 0 Warnings: 0
ALTER TABLE ...	Records: 3 Duplicates: 0 Warnings: 0
UPDATE ...	Rows matched: 40 Changed: 40 Warnings: 0

Nota: La consultas que no sean una de las antes mencionadas no están soportadas. En esta situación, **mysqli_info()** regresará una cadena vacía.

Valores retornados

Una cadena representando información adicional acerca de la consulta más recientemente ejecutada.

Ver también

mysqli_affected_rows(), **mysqli_warning_count()**, y **mysqli_num_rows()**.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$mysqli->query("CREATE TEMPORARY TABLE t1 LIKE City");

/* INSERT INTO .. SELECT */
$mysqli->query("INSERT INTO t1 SELECT * FROM City ORDER BY ID LIMIT 150");
printf("%s\n", $mysqli->info);

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

mysqli_query($link, "CREATE TEMPORARY TABLE t1 LIKE City");

/* INSERT INTO .. SELECT */
mysqli_query($link, "INSERT INTO t1 SELECT * FROM City ORDER BY ID LIMIT 150");
printf("%s\n", mysqli_info($link));

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Records: 150  Duplicates: 0  Warnings: 0
```

mysqli_init

(PHP 5)

mysqli_init -- Inicializa MySQLi y regresa un objeto para ser usado con mysqli_rea_connect

Descripción

mysqli **mysqli_init** (void)

Asigna o inicializa un objeto MySQL conveniente para **mysqli_options()** y **mysqli_real_connect()**.

> **Nota:** Cualquier llamada subsecuente a cualquier funció de mysqli (excepto **mysqli_options()**) fallará hasta que sea llamada **mysqli_real_connect()**.

Valores retornados

Regresa un objeto.

Ver también

mysqli_options(), **mysqli_close()**, **mysqli_real_connect()**, y **mysqli_connect()**.

mysqli_insert_id

(PHP 5)

mysqli_insert_id

(no version information, might be only in CVS)

mysqli->insert_id -- Regresa el ID generado automáticamente en la última consulta

Descripción

Estilo por procedimientos:

mixto **mysqli_insert_id** (mysqli identificador_de_enlace)

Estilo orientado a objetos (propiedad):

class **mysqli** {

mixto insert_id

}

La función **mysqli_insert_id()** regresa el ID generado por una consulta en una tabla con una columna que tiene el atributo AUTO_INCREMENT. Si la última consulta no fue un estatuto INSERT o UPDATE o si la tabla modificada no tiene una columna con este atributo, está función regresará cero.

> **Nota:** Ejecuta una sentencia INSERT o UPDATE usando la función LAST_INSERT_ID() también modificará el valor regresado por la función **mysqli_insert_id()**.

Valores retornados

El valor de el campo *AUTO_INCREMENT* que fue actualizado por la consulta previa. Regresa cero si no hubo una consulta previa en la conexión o si la consulta no actualizo un valor *AUTO_INCREMENT*.

> **Nota:** Si el número es mayor al entero máximo, **mysqli_insert_id()** regresará una cadena.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$mysqli->query("CREATE TABLE myCity LIKE City");

$query = "INSERT INTO myCity VALUES (NULL, 'Stuttgart', 'DEU', 'Stuttgart', 617000)";
$mysqli->query($query);

printf ("New Record has id %d.\n", $mysqli->insert_id);

/* drop table */
$mysqli->query("DROP TABLE myCity");

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

mysqli_query($link, "CREATE TABLE myCity LIKE City");

$query = "INSERT INTO myCity VALUES (NULL, 'Stuttgart', 'DEU', 'Stuttgart', 617000)";
mysqli_query($link, $query);

printf ("New Record has id %d.\n", mysqli_insert_id($link));

/* drop table */
mysqli_query($link, "DROP TABLE myCity");

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
New Record has id 1.
```

mysqli_kill

(PHP 5)

mysqli_kill

(no version information, might be only in CVS)

mysqli->kill -- Le pide al servidor "matar" el proceso MySQL

Descripción

Estilo por procedimientos:

bool **mysqli_kill** (mysqli identificador_de_enlace, int processid)

Estilo orientado a objetos (métod0)

class **mysqli** {

bool **kill** (int processid)

}

Esta función es usada para pedir al servidor matar le proceso MySQL específicado por el parámetro *processid*. Este valor debe ser obtenido llamando la función **mysqli_thread_id()**.

> **Nota:** Para detenener una consulta que se está ejecutando, debes usar el commando *KILL QUERY processid*.

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Ver también

mysqli_thread_id().

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

/* determine our thread id */
$thread_id = $mysqli->thread_id;

/* Kill connection */
$mysqli->kill($thread_id);

/* This should produce an error */
if (!$mysqli->query("CREATE TABLE myCity LIKE City")) {
    printf("Error: %s\n", $mysqli->error);
    exit;
}

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

/* determine our thread id */
$thread_id = mysqli_thread_id($link);

/* Kill connection */
mysqli_kill($link, $thread_id);

/* This should produce an error */
if (!mysqli_query($link, "CREATE TABLE myCity LIKE City")) {
    printf("Error: %s\n", mysqli_error($link));
    exit;
}

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Error: MySQL server has gone away
```

mysqli_master_query

(PHP 5)

mysqli_master_query -- Hace cumplir la ejecución de una consulta en el servidor en una configuración cliente/servidor

Descripción

bool **mysqli_master_query** (mysqli identificador_de_enlace, cadena consulta)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

mysqli_more_results

(PHP 5)

mysqli_more_results

(no version information, might be only in CVS)

mysqli->more_results -- Checa si hay má resultados de consultas de una consulta múltiple

Descripción

bool **mysqli_more_results** (mysqli identificador_de_enlace)

mysqli_more_results() indica si uno o más resultados están disponibles de una llamada previa a **mysqli_multi_query()**.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ver también

mysqli_multi_query(), **mysqli_next_result()**, **mysqli_store_result()**, y **mysqli_use_result()**.

Ejemplos

Vea **mysqli_multi_query()**.

mysqli_multi_query

(PHP 5)

mysqli_multi_query

(no version information, might be only in CVS)

mysqli->multi_query -- Ejecuta una consulta en la base de datos

Descripción

Estilo por procedimientos:

bool **mysqli_multi_query** (mysqli identificador_de_enlace, cadena consulta)

Estilo orientado a objetos (método):

class **mysqli** {

bool **multi_query** (cadena consulta)

}

La función **mysqli_multi_query()** ejecuta una o múltiples consultas las cuales estan concatenadas por un punto y coma.

Para obtener los resultados de la primera consulta tu puedes usar **mysqli_use_result()** o **mysqli_store_result()**. Todas las consultas subsecuentes pueden ser procesadas usando **mysqli_more_results()** y **mysqli_next_result()**.

Valores retornados

mysqli_multi_query() solo regresa **`FALSE`** si falla la primera sentencia. Para obtener los errores subsecuentes de las otras sentencias tiene que llamar primero **mysqli_next_result()**.

Ver también

mysqli_use_result(), **mysqli_store_result()**, **mysqli_next_result()**, y **mysqli_more_results()**.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$query  = "SELECT CURRENT_USER();";
$query .= "SELECT Name FROM City ORDER BY ID LIMIT 20, 5";

/* execute multi query */
if ($mysqli->multi_query($query)) {
    do {
        /* store first result set */
        if ($result = $mysqli->store_result()) {
            while ($row = $result->fetch_row()) {
                printf("%s\n", $row[0]);
            }
            $result->close();
        }
        /* print divider */
        if ($mysqli->more_results()) {
            printf("-----------------\n");
        }
    } while ($mysqli->next_result());
}

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$query  = "SELECT CURRENT_USER();";
$query .= "SELECT Name FROM City ORDER BY ID LIMIT 20, 5";

/* execute multi query */
if (mysqli_multi_query($link, $query)) {
    do {
        /* store first result set */
        if ($result = mysqli_store_result($link)) {
            while ($row = mysqli_fetch_row($result)) {
                printf("%s\n", $row[0]);
            }
            mysqli_free_result($result);
        }
        /* print divider */
        if (mysqli_more_results($link)) {
            printf("-----------------\n");
        }
    } while (mysqli_next_result($link));
}

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
my_user@localhost
-----------------
Amersfoort
Maastricht
Dordrecht
Leiden
Haarlemmermeer
```

mysqli_next_result

(PHP 5)

mysqli_next_result

(no version information, might be only in CVS)

mysqli->next_result -- Prepara el siguiente resultado de una consulta múltiple

Descripción

bool **mysqli_next_result** (mysqli identificador_de_enlace)

La función **mysqli_next_result()** prepara el siguiente resultado de una llamada previa a **mysqli_multi_query()** el cuál puede ser obtenido por **mysqli_store_result()** o **mysqli_use_result()**.

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Ver también

mysqli_multi_query(), **mysqli_more_results()**, **mysqli_store_result()**, y **mysqli_use_result()**.

Ejemplos

Vea **mysqli_multi_query()**.

mysqli_num_fields

(PHP 5)

mysqli_num_fields

(no version information, might be only in CVS)

result->field_count -- Obtiene el número de campos en un resultado Get the number of fields in a result

Descripción

Estilo por procedimientos:

int **mysqli_num_fields** (mysqli_result resultado)

Estilo orientado a objetos (propiedad

class **mysqli_result** {

int field_count

}

mysqli_num_fields() regresa el número de campos del resultado específicado.

Valores retornados

El númerp de campos en un resultado

Ver también

mysqli_fetch_field().

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

if ($result = $mysqli->query("SELECT * FROM City ORDER BY ID LIMIT 1")) {

    /* determine number of fields in result set */
    $field_cnt = $result->field_count;

    printf("Result set has %d fields.\n", $field_cnt);

    /* close result set */
    $result->close();
}

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

if ($result = mysqli_query($link, "SELECT * FROM City ORDER BY ID LIMIT 1")) {

    /* determine number of fields in result set */
    $field_cnt = mysqli_num_fields($result);

    printf("Result set has %d fields.\n", $field_cnt);

    /* close result set */
    mysqli_free_result($result);
}

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Result set has 5 fields.
```

mysqli_num_rows

(PHP 5)

mysqli_num_rows -- Obtiene el número de filas en un resultado

Descripción

Estilo por procedimientos:

mixto **mysqli_num_rows** (mysqli_result resultado)

Estilo orientado a objetos (propiedad):

class **mysqli_result** {

mixto num_rows

}

Regresa el número de filas en el resultado.

El uso de **mysqli_num_rows()** depende de si se usa resultados con almacenamiento intermedio (buffered) o no. En caso de usar resultados sin almacenamiento intermedio, **mysqli_num_rows()** no dará el número correcto de filas hasta que se hayan obtenido todas las filas del resultado.

Valores retornados

Regresa el número de filas en el resultado.

> **Nota:** Si el número de filas es superior al valor entero maximo, el número será regresado como una cadena.

Ver también

mysqli_affected_rows(), mysqli_store_result(), mysqli_use_result(), y mysqli_query().

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

if ($result = $mysqli->query("SELECT Code, Name FROM Country ORDER BY Name")) {

    /* determine number of rows result set */
    $row_cnt = $result->num_rows;

    printf("Result set has %d rows.\n", $row_cnt);

    /* close result set */
    $result->close();
}

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

if ($result = mysqli_query($link, "SELECT Code, Name FROM Country ORDER BY Name")) {

    /* determine number of rows result set */
    $row_cnt = mysqli_num_rows($result);

    printf("Result set has %d rows.\n", $row_cnt);

    /* close result set */
    mysqli_free_result($result);
}

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Result set has 239 rows.
```

mysqli_options

(PHP 5)

mysqli_options

(no version information, might be only in CVS)

mysqli->options -- Fija opciones

Descripción

Estilo por procedimientos:

bool **mysqli_options** (mysqli identificador_de_enlace, int opción, mixto valor)

Estilo orientado a objetos (método)

class **mysqli** {

bool **options** (int opción, mixto valor)

}

mysqli_options() puede ser usada para fijar opciones extra en la conexión y afectar el comportamiento para la conexión.

Esta función puede ser llamada múltiples veces para fijar diferentes opciones.

mysqli_options() debe ser llamada después de **mysqli_init()** y antes de **mysqli_real_connect()**.

El parámetro *opción* es la opción que se busca cambiar, el parámetro *valor* es el valor para la opción. El parámetro *opciÃ‚Â´on* puede ser una de las siguientes:

Tabla 1. Opciones validas

Nombre	Descripción
MYSQLI_OPT_CONNECT_TIMEOUT	Tiempo que espera para ser conectado en segundos
MYSQLI_OPT_LOCAL_INFILE	habilita/deshabilita el uso de *LOAD LOCAL INFILE*
MYSQLI_INIT_CMD	comando a ejecutar después cuando se conecte al servidor MySQL
MYSQLI_READ_DEFAULT_FILE	Lee las opciones del archivo de configuración específicado en vez de`my.cnf`
MYSQLI_READ_DEFAULT_GROUP	Lee las opciones del grupo del archivo`my.cnf` o del archivo específicado con **MYSQL_READ_DEFAULT_FILE**.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ver también

mysqli_init(), y **mysqli_real_connect()**.

Ejemplos

Vea **mysqli_real_connect()**.

mysqli_param_count

mysqli_param_count -- Alias para **mysqli_stmt_param_count()**

Descripción

Esta función es un alias de **mysqli_stmt_param_count()**. Para una descipción detallada vea **mysqli_stmt_param_count()**.

> **Nota: mysqli_param_count()** es obsoleta y será removida.

Ver también

mysqli_stmt_param_count().

mysqli_ping

(PHP 5)

mysqli_ping

(no version information, might be only in CVS)

mysqli->ping -- Revisa una conexión al servidor o intenta reconectar si la conexión se ha perdido

Descripción

Estilo por procedimientos:

bool **mysqli_ping** (mysqli identificador_de_enlace)

Estilo orientado a objetos (método):

class **mysqli** {

bool **ping** (void)

}

Checa si la conexión al servidor está operando. Si se perdiÃ‚Â´ y la opción global *mysqli.reconnect* esta activa se intenta una reconexión automática.

Esta función puede ser usada por clientes que duran mucho tiempo inactivos, para checar si el servidor ha verrado al conexión y reconectarse en caso necesario.

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

/* check if server is alive */
if ($mysqli->ping()) {
    printf ("Our connection is ok!\n");
} else {
    printf ("Error: %s\n", $mysqli->error);
}

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

/* check if server is alive */
if (mysqli_ping($link)) {
    printf ("Our connection is ok!\n");
} else {
    printf ("Error: %s\n", mysqli_error($link));
}

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Our connection is ok!
```

mysqli_prepare

(PHP 5)

mysqli_prepare

(no version information, might be only in CVS)

mysqli->prepare -- Prepara una sentencia SQL para su ejecución

Descripción

Estilo por procedimientos:

mixto **mysqli_prepare** (mysqli identificador_de_enlace, cadena query)

Estilo orientado a objetos (método)

class **mysqli** {

mixto **prepare** (cadena query)

}

mysqli_prepare() prepara la consulta SQL acentuada por la terminación en caracter nulo, y regresa un manejador para ser usado para las operaciones en la sentencia. La consulta debe consistir de un solo enunciado SQL.

> **Nota:** No debe agregar el punto y coma al fina o *g* al enunciado.

El parámetro *query* puede incluir uno o más marcadores de parámetros en la sentencia SQL, incluyendo el caracter (*?*) en las posiciones apropiadas.

Nota: Los marcadores son legales solo en ciertos lugares de la sentencia SQL. Por ejemplo, son permitidos en la lista de VALUES() de una sentencia INSERT (para específicar los valores de las columnas en la fila), o en una comparación con una columna en una clausula WHERE para especificar un valor de comparación.

Sin embargo, no son permitidos para identificadores (tales como nombres de tablas y columnas, en la selección de los nombres de las columnas a ser regresadas por la sentencia SELECT), o para especificar operadores binarios tales como = el signo de igual. La última restricción es necesaria porque sería imposible determinar el tipo del parámetro. En general, los parámetros son legales solo en las sentencias del lenguaje de manipulación de datos (DML), y no en las sentencias del lenguaje de definición de datos (DDL).

Los marcadores de parámetros deben estar ligados a variables de la aplicación usando **mysqli_stmt_bind_param()** y/o **mysqli_stmt_bind_result()** antes de ejecutar la sentencia SQL u obtener las filas.

Valores retornados

mysqli_prepare() regresa un objeto o **`FALSE`** si ocurre un erro.

Ver también

mysqli_stmt_execute(), **mysqli_stmt_fetch()**, **mysqli_stmt_bind_param()**, **mysqli_stmt_bind_result()**>, y **mysqli_stmt_close()**.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```php
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$city = "Amersfoort";

/* create a prepared statement */
if ($stmt = $mysqli->prepare("SELECT District FROM City WHERE Name=?")) {

    /* bind parameters for markers */
    $stmt->bind_param("s", $city);

    /* execute query */
    $stmt->execute();

    /* bind result variables */
    $stmt->bind_result($district);

    /* fetch value */
    $stmt->fetch();

    printf("%s is in district %s\n", $city, $district);

    /* close statement */
    $stmt->close();
}

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$city = "Amersfoort";

/* create a prepared statement */
if ($stmt = mysqli_prepare($link, "SELECT District FROM City WHERE Name=?")) {

    /* bind parameters for markers */
    mysqli_stmt_bind_param($stmt, "s", $city);

    /* execute query */
    mysqli_stmt_execute($stmt);

    /* bind result variables */
    mysqli_stmt_bind_result($stmt, $district);

    /* fetch value */
    mysqli_stmt_fetch($stmt);

    printf("%s is in district %s\n", $city, $district);

    /* close statement */
    mysqli_stmt_close($stmt);
}

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Amersfoort is in district Utrecht
```

mysqli_query

(PHP 5)

mysqli_query

(no version information, might be only in CVS)

mysqli->query -- Ejecuta una consulta en la base de datos

Descripción

Estilo por procedimientos:

mixto **mysqli_query** (mysqli identificador_de_enlace, cadena consulta [, int resultmode])

Estilo orientado a objetos (método):

class **mysqli** {

mixto **query** (cadena consulta)

}

La funció **mysqli_query()** es usada para simplificar la acción de ejecutar una consulta sobre la base de datos representada por el parámetro *identificador_de_enlace*.

Funcionalmente, usar está función es identifo a llamar **mysqli_real_query()** seguida ya sea de **mysqli_use_result()** o **mysqli_store_result()** donde *consulta* es la cadena de la consulta y *resultmode* es una de las constantes *MYSQLI_USE_RESULT* o *MYSQLI_STORE_RESULT* dependiendo del comportamiento deseado. Por defecto, si no se da *resultmode*, *MYSQLI_STORE_RESULT* es usado.

Si usted ejecuta **mysqli_query()** con *resultmode MYSQLI_USE_RESULT* todas las llamadas subsecuentes regresarán el código de error *Commands out of sync* a menos que llame **mysqli_free_result()**.

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo. Para *SELECT, SHOW, DESCRIBE* o *EXPLAIN* **mysqli_query()** regresará un objeto resultante.

Ver también

mysqli_real_query(), **mysqli_multi_query()**, y **mysqli_free_result()**.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

/* Create table doesn't return a resultset */
if ($mysqli->query("CREATE TEMPORARY TABLE myCity LIKE City") === TRUE) {
    printf("Table myCity successfully created.\n");
}

/* Select queries return a resultset */
if ($result = $mysqli->query("SELECT Name FROM City LIMIT 10")) {
    printf("Select returned %d rows.\n", $result->num_rows);

    /* free result set */
    $result->close();
}

/* If we have to retrieve large amount of data we use MYSQLI_USE_RESULT */
if ($result = $mysqli->query("SELECT * FROM City", MYSQLI_USE_RESULT)) {

    /* Note, that we can't execute any functions which interact with the
       server until result set was closed. All calls will return an
       'out of sync' error */
    if (!$mysqli->query("SET @a:='this will not work'")) {
        printf("Error: %s\n", $mysqli->error);
    }
    $result->close();
}

$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

/* Create table doesn't return a resultset */
if (mysqli_query($link, "CREATE TEMPORARY TABLE myCity LIKE City") === TRUE) {
    printf("Table myCity successfully created.\n");
}

/* Select queries return a resultset */
if ($result = mysqli_query($link, "SELECT Name FROM City LIMIT 10")) {
    printf("Select returned %d rows.\n", mysqli_num_rows($result));

    /* free result set */
    mysqli_free_result($result);
}

/* If we have to retrieve large amount of data we use MYSQLI_USE_RESULT */
if ($result = mysqli_query($link, "SELECT * FROM City", MYSQLI_USE_RESULT)) {

    /* Note, that we can't execute any functions which interact with the
       server until result set was closed. All calls will return an
       'out of sync' error */
    if (!mysqli_query($link, "SET @a:='this will not work'")) {
        printf("Error: %s\n", mysqli_error($link));
    }
    mysqli_free_result($result);
}

mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Table myCity successfully created.
Select returned 10 rows.
Error: Commands out of sync;  You can't run this command now
```

mysqli_real_connect

(PHP 5)

mysqli_real_connect

(no version information, might be only in CVS)

mysqli->real_connect -- Abre una conexión a un servidor de MySQL

Descripción

Estilo por procedimientos

bool **mysqli_real_connect** (mysqli link [, cadena equipo_huésped [, cadena usuario [, cadena contraseña [, cadena nombre_de_bd [, int puerto [, cadena socket [, int banderas]]]]]]])

Estilo orientado a objetos (método)

class **mysqli** {

bool **real_connect** ([cadena equipo_huésped [, cadena usuario [, cadena contraseña [, cadena nombre_de_bd [, int puerto [, cadena socket [, int banderas]]]]]]])

}

mysqli_real_connect() Intenta establecer una conexión con un equipo que tiene ejecutando un servidor de MySQL representado por el parámetro *equipo_huésped*.

Esta función es diferente de **mysqli_connect()** en lo siguiente:

- **mysqli_real_connect()** necesita un objeto valido el cual tiene que ser creado por la función **mysqli_init()**
- Con la función **mysqli_options()** tu puedes fijar varias opciones por conexión.
- Con el parámetro *banderas* tu puedes fijar diferentes opciones de la conexión:

  Tabla 1. Banderas soportadas

Nombre	Descripción
MYSQLI_CLIENT_COMPRESS	Usa el protocolo de compresión
MYSQLI_CLIENT_FOUND_ROWS	Regresa el número de filas encontradas, no el número de filas afectadas
MYSQLI_CLIENT_IGNORE_SPACE	Permite espacios despues de los nombres de las funciones. Hace palabras reservadas los nombres de las funciones.
MYSQLI_CLIENT_INTERACTIVE	Permite los segundos *interactive_timeout* (en vez de *wait_timeout*) de inactividad antes de cerrar la conexión
MYSQLI_CLIENT_SSL	Usa SSL (encriptación)

  Nota: Por razones de seguridad la bandera *MULTI_STATEMENT* no es permitida en PHP. Si quiere ejecutar consultas múltiples use la función **mysqli_multi_query()**.

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Ver también

mysqli_connect(), **mysqli_init()**, **mysqli_options()**, **mysqli_ssl_set()**, y **mysqli_close()**.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```php
<?php

/* create a connection object which is not connected */
$mysqli = mysqli_init();

/* set connection options */
$mysqli->options(MYSQLI_INIT_COMMAND, "SET AUTOCOMMIT=0");
$mysqli->options(MYSQLI_OPT_CONNECT_TIMEOUT, 5);

/* connect to server */
$mysqli->real_connect('localhost', 'my_user', 'my_password', 'world');

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

printf ("Connection: %s\n.", $mysqli->host_info);

$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```php
<?php

/* create a connection object which is not connected */
$link = mysqli_init();

/* set connection options */
mysqli_options($link, MYSQLI_INIT_COMMAND, "SET AUTOCOMMIT=0");
mysqli_options($link, MYSQLI_OPT_CONNECT_TIMEOUT, 5);

/* connect to server */
mysqli_real_connect($link, 'localhost', 'my_user', 'my_password', 'world');

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

printf ("Connection: %s\n.", mysqli_get_host_info($link));

mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Connection: Localhost via UNIX socket
```

mysqli_real_escape_string

(PHP 5)

mysqli_real_escape_string

(no version information, might be only in CVS)

mysqli->real_escape_string -- Protege caracteres especiales en una cadena para ser usada en una sentencia SQL, tomando en cuenta el conjunto de caracteres para la conexión

Descripción

Estilo por procedimientos:

cadena **mysqli_real_escape_string** (mysqli identificador_de_enlace, cadena escapestr)

Estilo orientado a objetos (método):

class **mysqli** {

cadena **real_escape_string** (cadena escapestr)

}

Esta función es usada para crear una cadena SQL legal que se puede usar en una sentencia SQL. La cadena *escapestr* está codificada para una cadena SQL protegida, tomando en cuenta el conjunto de caracteres actual para la conexión.

Los caracteres codificados son *NUL (ASCII 0), \n, \r, \, ', ", y Control-Z.*

Valores retornados

Regresa una cadena protegida.

Ver también

mysqli_character_set_name().

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$mysqli->query("CREATE TEMPORARY TABLE myCity LIKE City");

$city = "'s Hertogenbosch";

/* this query will fail, cause we didn't escape $city */
if (!$mysqli->query("INSERT into myCity (Name) VALUES ('$city')")) {
    printf("Error: %s\n", $mysqli->sqlstate);
}

$city = $mysqli->real_escape_string($city);

/* this query with escaped $city will work */
if ($mysqli->query("INSERT into myCity (Name) VALUES ('$city')")) {
    printf("%d Row inserted.\n", $mysqli->affected_rows);
}

$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

mysqli_query($link, "CREATE TEMPORARY TABLE myCity LIKE City");

$city = "'s Hertogenbosch";

/* this query will fail, cause we didn't escape $city */
if (!mysqli_query($link, "INSERT into myCity (Name) VALUES ('$city')")) {
    printf("Error: %s\n", mysqli_sqlstate($link));
}

$city = mysqli_real_escape_string($link, $city);

/* this query with escaped $city will work */
if (mysqli_query($link, "INSERT into myCity (Name) VALUES ('$city')")) {
    printf("%d Row inserted.\n", mysqli_affected_rows($link));
}

mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Error: 42000
1 Row inserted.
```

mysqli_real_query

(PHP 5)

mysqli_real_query

(no version information, might be only in CVS)

mysqli->real_query -- Ejecuta una consulta SQL

Descripción

Estilo por procedimientos

bool **mysqli_real_query** (mysqli identificador_de_enlace, cadena consulta)

Estilo orientado a objetos (método):

class **mysqli** {

bool **real_query** (cadena consulta)

}

La funciÃƒâ€œn **mysqli_real_query()** es usada para ejecutar solo una consulta sobre la base de

datos representada por *identificador_de_enlace* cuyo resultado puede ser obtenido o almacenado usando las funciones **mysqli_store_result()** o **mysqli_use_result()**.

> **Nota:** Para determinar si una consulta data debe regresar un resultado o no, vea **mysqli_field_count()**.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ver también

mysqli_query(), **mysqli_store_result()**, y **mysqli_use_result()**.

mysqli_report

(PHP 5)

mysqli_report -- Habilita o deshabilita las funciones internas de reporte

Descripción

bool **mysqli_report** (int banderas)

mysqli_report() es una poderosa función para mejorar tus consultas y código durante las etapas de desarrollo y prueba. Dependiendo de las banderas reporta errores de las llamadas a las funciones o las consultas que no usan un índice (o usa un mal índice).

Tabla 1. Banderas permitidas

Nombre	Descripción
MYSQLI_REPORT_OFF	Deshabilita el reporteado
MYSQLI_REPORT_ERROR	Reporta errores de las llamadas a cualquier función
MYSQLI_REPORT_INDEX	Reporta si no se usa índice o si se usó un índice mal construido en una consulta
MYSQLI_REPORT_ALL	Fija todas las opciones (reporta todo)

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ver también

mysqli_debug(), y **mysqli_dump_debug_info()**.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
/* activate reporting */
mysqli_report(MYSQLI_REPORT_ALL);

$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

/* this query should report an error */
$result = $mysqli->query("SELECT Name FROM Nonexistingtable WHERE population > 50000");

/* this query should report a warning */
$result = $mysqli->query("SELECT Name FROM City WHERE population > 50000");
$result->close();

$mysqli->close();
?>
```

mysqli_rollback

(PHP 5)

mysqli_rollback

(no version information, might be only in CVS)

mysqli->rollback -- Regresa/deshace la transacción actual

Descripción

bool **mysqli_rollback** (mysqli identificador_de_enlace)

class **mysqli** {

bool **rollback** (void)

}

Regresa o deshace la transacción actual pra la base de datos específicada por el parámetro *identificador_de_enlace*.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ver también

mysqli_commit(), y mysqli_autocommit().

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

/* disable autocommit */
$mysqli->autocommit(FALSE);

$mysqli->query("CREATE TABLE myCity LIKE City");
$mysqli->query("ALTER TABLE myCity Type=InnoDB");
$mysqli->query("INSERT INTO myCity SELECT * FROM City LIMIT 50");

/* commit insert */
$mysqli->commit();

/* delete all rows */
$mysqli->query("DELETE FROM myCity");

if ($result = $mysqli->query("SELECT COUNT(*) FROM myCity")) {
    $row = $result->fetch_row();
    printf("%d rows in table myCity.\n", $row[0]);
    /* Free result */
    $result->close();
}

/* Rollback */
$mysqli->rollback();

if ($result = $mysqli->query("SELECT COUNT(*) FROM myCity")) {
    $row = $result->fetch_row();
    printf("%d rows in table myCity (after rollback).\n", $row[0]);
    /* Free result */
    $result->close();
}

/* Drop table myCity */
$mysqli->query("DROP TABLE myCity");

$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```php
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

/* disable autocommit */
mysqli_autocommit($link, FALSE);

mysqli_query($link, "CREATE TABLE myCity LIKE City");
mysqli_query($link, "ALTER TABLE myCity Type=InnoDB");
mysqli_query($link, "INSERT INTO myCity SELECT * FROM City LIMIT 50");

/* commit insert */
mysqli_commit($link);

/* delete all rows */
mysqli_query($link, "DELETE FROM myCity");

if ($result = mysqli_query($link, "SELECT COUNT(*) FROM myCity")) {
    $row = mysqli_fetch_row($result);
    printf("%d rows in table myCity.\n", $row[0]);
    /* Free result */
    mysqli_free_result($result);
}

/* Rollback */
mysqli_rollback($link);

if ($result = mysqli_query($link, "SELECT COUNT(*) FROM myCity")) {
    $row = mysqli_fetch_row($result);
    printf("%d rows in table myCity (after rollback).\n", $row[0]);
    /* Free result */
    mysqli_free_result($result);
}

/* Drop table myCity */
mysqli_query($link, "DROP TABLE myCity");

mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
0 rows in table myCity.
50 rows in table myCity (after rollback).
```

mysqli_rpl_parse_enabled

(PHP 5)

mysqli_rpl_parse_enabled -- Revisa si el analizador RPL está habilitado

Descripción

int **mysqli_rpl_parse_enabled** (mysqli identificador_de_enlace)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

mysqli_rpl_probe

(PHP 5)

mysqli_rpl_probe -- Prueba RPL

Descripción

bool **mysqli_rpl_probe** (mysqli identificador_de_enlace)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

mysqli_rpl_query_type

(PHP 5)

mysqli_rpl_query_type

(no version information, might be only in CVS)

mysqli->rpl_query_type -- Regresa el tipo de consulta RPL

Descripción

Estilo por procedimientos:

int **mysqli_rpl_query_type** (mysqli link, string query)

Estilo orientado a objetos (método)

class **mysqli** {

int **rpl_query_type** (string query)

}

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

mysqli_select_db

(PHP 5)

mysqli_select_db

(no version information, might be only in CVS)

mysqli->select_db -- Selecciona la base de datos por defecto para las consultas de la base de datos

Descripción

bool **mysqli_select_db** (mysqli identificador_de_enlace, cadena dbname)

La función **mysqli_select_db()** selecciona la base de datos por defecto (específicada por el parámetro *dbname*) a ser usada cuando se ejecuten consultas sobre la conexión de base de datos representada por el parámetro *identificador_de_enlace* parameter.

> **Nota:** Esta función solo puede ser usada para cambiar la base de datos por defecto para la conexión. Usted puede seleccionar la base de datos en el cuarto parámetro en **mysqli_connect()**.

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Ver también

mysqli_connect(), y **mysqli_real_connect()**.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```php
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "test");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

/* return name of current default database */
if ($result = $mysqli->query("SELECT DATABASE()")) {
    $row = $result->fetch_row();
    printf("Default database is %s.\n", $row[0]);
    $result->close();
}

/* change db to world db */
$mysqli->select_db("world");

/* return name of current default database */
if ($result = $mysqli->query("SELECT DATABASE()")) {
    $row = $result->fetch_row();
    printf("Default database is %s.\n", $row[0]);
    $result->close();
}

$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```php
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "test");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

/* return name of current default database */
if ($result = mysqli_query($link, "SELECT DATABASE()")) {
    $row = mysqli_fetch_row($result);
    printf("Default database is %s.\n", $row[0]);
    mysqli_free_result($result);
}

/* change db to world db */
mysqli_select_db($link, "world");

/* return name of current default database */
if ($result = mysqli_query($link, "SELECT DATABASE()")) {
    $row = mysqli_fetch_row($result);
    printf("Default database is %s.\n", $row[0]);
    mysqli_free_result($result);
}

mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Default database is test.
Default database is world.
```

mysqli_send_long_data

mysqli_send_long_data -- Alias para **mysqli_stmt_send_long_data()**

Descripción

Esta funció es un alias de **mysqli_stmt_send_long_data()**. Para una descripción detallada vea **mysqli_stmt_send_long_data()**.

> **Nota: mysqli_send_long_data()** es obsoleta y será removida.

Ver también

mysqli_stmt_send_long_data().

mysqli_send_query

(PHP 5)

mysqli_send_query

(no version information, might be only in CVS)

mysqli->send_query -- Envía la consulta y regresa

Description

Procedural style:

bool **mysqli_send_query** (mysqli link, string query)

Object oriented style (method)

class **mysqli** {

bool **send_query** (string query)

}

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

mysqli_server_end

(PHP 5)

mysqli_server_end -- Termina el proceso del Servidor

Descripción

void **mysqli_server_end** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

mysqli_server_init

(PHP 5)

mysqli_server_init -- Inicializa un servidor embebido

Descripción

bool **mysqli_server_init** ([array server [, array groups]])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

mysqli_set_opt

mysqli_set_opt -- Alias de **mysqli_options()**

Descripción

Esta función es un alias de **mysqli_options()**.

mysqli_sqlstate

(PHP 5)

mysqli_sqlstate

(no version information, might be only in CVS)

mysqli->sqlstate -- Regresa el error SQLSTATE de la operación MySQL previa

Descripción

Estilo por procedimientos:

cadena **mysqli_sqlstate** (mysqli identificador_de_enlace)

Estilo orientado a objetos (propiedad):

class **mysqli** {

int sqlstate

}

Regresa una cadena conteniendo el código de error SQLSTATE para el último error. El código de error consiste en cinco caracteres. *'00000'* significa que no hay error. Los valores están especificados por el SQL ANSI y ODBC. Para una lista de los valores posibles, vea http://dev.mysql.com/doc/mysql/en/Error-returns.html.

> **Nota:** Noete que no todos los errores de MySQL han sido mapeados a los de SQLSTATE. El valor *HY000* (error general) es usado para errores no mapeados.

Valores retornados

Regresa una cadena conteniendo el código de error SQLSTAT para el último error. El código de error consiste en cinco caracteres. *'00000'* significa que no hubo error.

Ver también

mysqli_errno(), y **mysqli_error()**.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

/* Table City already exists, so we should get an error */
if (!$mysqli->query("CREATE TABLE City (ID INT, Name VARCHAR(30))")) {
    printf("Error - SQLSTATE %s.\n", $mysqli->sqlstate);
}

$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

/* Table City already exists, so we should get an error */
if (!mysqli_query($link, "CREATE TABLE City (ID INT, Name VARCHAR(30))")) {
    printf("Error - SQLSTATE %s.\n", mysqli_sqlstate($link));
}

mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Error - SQLSTATE 42S01.
```

mysqli_ssl_set

(PHP 5)

mysqli_ssl_set

(no version information, might be only in CVS)

mysqli->ssl_set -- Usado para establecer conexiones seguras usando SSL

Descripción

Estilo por procedimientos:

bool **mysqli_ssl_set** (mysqli identificador_de_enlace, cadena key, cadena cert, cadena ca, cadena capath, cadena cipher)

Estilo orientado a objetos (método):

class **mysqli** {

bool **ssl_set** (cadena key, cadena cert, cadena ca, cadena capath, cadena cipher)

}

La función **mysqli_ssl_set()** es usada para establecer conexiones seguras usando SSL. Esta debe ser llamada antes de **mysqli_real_connect()**. Esta función no hace nada a menos que el soporte OpenSSL este activo.

key es la ubicación del archivo principal. *cert* es la ubicación del archivo del certificado. *ca* es la ubicación del archivo de autorización del certificado. *capath* es la ubicación del directorio que contiene los certificados SSL CA en que se confí en formato pem. *cipher* es una lista de los ciphes permitidos a usar para la encriptación SSL. Cualquier parámetro SSL sin usar, recibirá el valor **NULL**

Valores retornados

Está función siempre regresa el valor **TRUE**. Si la disposición de SSL está incorrecta **mysqli_real_connect()** regresará un error cuando se intente conectar.

Ver también

mysqli_options(), y **mysqli_real_connect()**.

mysqli_stat

(PHP 5)

mysqli_stat

(no version information, might be only in CVS)

mysqli->stat -- Obtiene el estado actual del sistema

Descripción

Estilo por procedimientos:

mixto **mysqli_stat** (mysqli identificador_de_enlace)

Estilo orientado a objetos (método):

class **mysqli** {

mixto **stat** (void)

}

mysqli_stat() regresa una cadena conteniendo información similar a la proveida por el comando 'mysqladmin status'. Esto incluye el tiempo total activo del servidor en segundos y en número de procesos ejecutandose, preguntas, recargas y tablas abiertas.

Valores retornados

Una cadena describiendo el estado del servidor. **FALSE** si un error ocurre.

Ver también

mysqli_get_server_info().

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

printf ("System status: %s\n", $mysqli->stat());

$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

printf("System status: %s\n", mysqli_stat($link));

mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
System status: Uptime: 272  Threads: 1  Questions: 5340  Slow queries: 0
Opens: 13  Flush tables: 1  Open tables: 0  Queries per second avg: 19.632
Memory in use: 8496K  Max memory used: 8560K
```

mysqli_stmt_affected_rows

(PHP 5)

mysqli_stmt_affected_rows

(no version information, might be only in CVS)

mysqli_stmt->affected_rows -- Regresa el número total de filas cambiadas, borradas o insertadas por la última sentencia ejecutada

Descripción

Estilo por procedimientos :

mixto **mysqli_stmt_affected_rows** (mysqli_stmt stmt)

EStilo orientado a objetos (propiedad):

class **mysqli_stmt** {

mixto affected_rows

}

mysqli_stmt_affected_rows() regresa el número de filas afectadas por consultas INSERT, UPDATE o DELETE. Si la última consulta fue invalida, esta función regresará -1

La función **mysqli_stmt_affected_rows()** solo funciona con consultas que actualizan tablas. Para obtener el número de filas de una consulta SELECT, use la función **mysqli_stmt_num_rows()**.

Valores retornados

Un entero mayor a cero indica el número de filas afectadas u obtenidas. Cero indica que no hubo registros actualizados para sentencias UPDATE/DELETE, no hubo filas que concordaran la cláusula WHERE en la consulta o que no se ha ejecutado ninguna consulta todavía. -1 indifca que la consulta ha regresado un error.

> **Nota:** Si el número de filas afectadas es mayor que el valor entero maximo de PHP, el número de filas afectadas será regresada como una cadena.

Ver también

mysqli_stmt_num_rows(), y **mysqli_prepare()**.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

/* create temp table */
$mysqli->query("CREATE TEMPORARY TABLE myCountry LIKE Country");

$query = "INSERT INTO myCountry SELECT * FROM Country WHERE Code LIKE ?";

/* prepare statement */
if ($stmt = $mysqli->prepare($query)) {

    /* Bind variable for placeholder */
    $code = 'A%';
    $stmt->bind_param("s", $code);

    /* execute statement */
    $stmt->execute();

          printf("rows inserted: %d\n", $stmt->affected_rows);

    /* close statement */
    $stmt->close();
}

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

/* create temp table */
mysqli_query($link, "CREATE TEMPORARY TABLE myCountry LIKE Country");

$query = "INSERT INTO myCountry SELECT * FROM Country WHERE Code LIKE ?";

/* prepare statement */
if ($stmt = mysqli_prepare($link, $query)) {

    /* Bind variable for placeholder */
    $code = 'A%';
    mysqli_stmt_bind_param($stmt, "s", $code);

    /* execute statement */
    mysqli_stmt_execute($stmt);

          printf("rows inserted: %d\n", mysqli_stmt_affected_rows($stmt));

    /* close statement */
    mysqli_stmt_close($stmt);
}

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
rows inserted: 17
```

mysqli_stmt_bind_param

(PHP 5)

mysqli_stmt_bind_param

(no version information, might be only in CVS)

stmt->bind_param -- Enlaza variables como parámetros a una sentencia preparada

Descripción

Estilo por procedimientos:

bool **mysqli_stmt_bind_param** (mysqli_stmt stmt, cadena types, mixto &var1 [, mixto & ...])

Estilo orientado a objetos (método):

class **mysqli_stmt** {

bool **bind_param** (matriz types, mixto &var1 [, mixto & ...])

}

mysqli_stmt_bind_param() es usada para enlazar variables para los marcadores de parámetros en la sentencia SQL que fue pasada a **mysqli_prepare()**. La cadena *types* contiene uno o más caracteres los cuales especifican los tipos para las variables enlazadas correspondientes.

Tabla 1. Especificación de caracteres de tipo

Caracter	Descripción
i	La variable correspondiente tiene tipo entero
d	La variable correspondiente tiene tipo doble
s	La variable correspondiente tiene tipo cadena
b	La variable correspondiente tiene tipo BLOB y será enviada en paquetes

Nota: Si el tamaño de los datos de la variable exceden el tamaño maximo permitido para un paquete (max_allowed_package), tienes que específicar *b* en *types* y usar **mysqli_stmt_send_long_data()** para enviar los datos en paquetes.

El número de variables y longitud de la cadena *types* debe coincidir los parámetros en la sentencia.

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Vea también

mysqli_stmt_bind_result(), **mysqli_stmt_execute()**, **mysqli_stmt_fetch()**, **mysqli_prepare()**, **mysqli_stmt_send_long_data()**, **mysqli_stmt_errno()**, y **mysqli_stmt_error()**.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli('localhost', 'my_user', 'my_password', 'world');

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$stmt = $mysqli->prepare("INSERT INTO CountryLanguage VALUES (?, ?, ?, ?)");
$stmt->bind_param('sssd', $code, $language, $official, $percent);

$code = 'DEU';
$language = 'Bavarian';
$official = "F";
$percent = 11.2;

/* execute prepared statement */
$stmt->execute();

printf("%d Row inserted.\n", $stmt->affected_rows);

/* close statement and connection */
$stmt->close();

/* Clean up table CountryLanguage */
$mysqli->query("DELETE FROM CountryLanguage WHERE Language='Bavarian'");
printf("%d Row deleted.\n", $mysqli->affected_rows);

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect('localhost', 'my_user', 'my_password', 'world');

/* check connection */
if (!$link) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$stmt = mysqli_prepare($link, "INSERT INTO CountryLanguage VALUES (?, ?, ?, ?)");
mysqli_stmt_bind_param($stmt, 'sssd', $code, $language, $official, $percent);

$code = 'DEU';
$language = 'Bavarian';
$official = "F";
$percent = 11.2;

/* execute prepared statement */
mysqi_stmt_execute($stmt);

printf("%d Row inserted.\n", mysqli_stmt_affected_rows($stmt));

/* close statement and connection */
mysqli_stmt_close($stmt);

/* Clean up table CountryLanguage */
mysqli_query($link, "DELETE FROM CountryLanguage WHERE Language='Bavarian'");
printf("%d Row deleted.\n", mysqli_affected_rows($link));

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
1 Row inserted.
1 Row deleted.
```

mysqli_stmt_bind_result

(PHP 5)

mysqli_stmt_bind_result

(no version information, might be only in CVS)

stmt->bind_result -- Enlaza variables a una sentencia preparada para el almacenamiento del resultado

Descripción

Estilo por procedimientos:

bool **mysqli_stmt_bind_result** (mysqli_stmt stmt, mixto &var1 [, mixto & ...])

Estilo orientado a objetos (método):

class **mysqli_stmt** {

bool **bind_result** (mixto &var1 [, mixto & ...])

}

mysqli_stmt_bind_result() es usada para asociar (enlazar) columnas en el resultado a variables. Cuando **mysqli_stmt_fetch()** es llamada para obtener datos, el protocolo cliente/servidor de MySQL coloca los datos para las columnas delimitadas en las variables específicadas en *var1, ...*

> **Nota:** Note que todas las columnas deben estar delimitadas antes de llamar **mysqli_stmt_fetch()**. Dependiendo en los tipos de columnas a las variables delimitadas pueden silenciosamente cambiar al tipo correspondiente en PHP.
>
> Una columna puede ser delimitada o re-delimitada en cualquier momento, aún despues de que se ha recuperado parcialmente un resultado. La nueva delimitación toma efecto la siguiente vez que **mysqli_stmt_fetch()** es llamada.

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Ver también

mysqli_stmt_bind_param(), **mysqli_stmt_execute()**, **mysqli_stmt_fetch()**, **mysqli_prepare()**, **mysqli_stmt_prepare()**, **mysqli_stmt_init()**, **mysqli_stmt_errno()**, y **mysqli_stmt_error()**.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

/* prepare statement */
if ($stmt = $mysqli->prepare("SELECT Code, Name FROM Country ORDER BY Name LIMIT 5")) {
    $stmt->execute();

    /* bind variables to prepared statement */
    $stmt->bind_result($col1, $col2);

    /* fetch values */
    while ($stmt->fetch()) {
        printf("%s %s\n", $col1, $col2);
    }

    /* close statement */
    $stmt->close();
}
/* close connection */
$mysqli->close();

?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (!$link) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

/* prepare statement */
if ($stmt = mysqli_prepare($link, "SELECT Code, Name FROM Country ORDER BY Name LIMIT 5
    mysqli_stmt_execute($stmt);

    /* bind variables to prepared statement */
    mysqli_stmt_bind_result($stmt, $col1, $col2);

    /* fetch values */
    while (mysqli_stmt_fetch($stmt)) {
       printf("%s %s\n", $col1, $col2);
    }

    /* close statement */
    mysqli_stmt_close($stmt);
}

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
AFG Afghanistan
ALB Albania
DZA Algeria
ASM American Samoa
AND Andorra
```

mysqli_stmt_close

(PHP 5)

mysqli_stmt_close

(no version information, might be only in CVS)

mysqli_stmt->close -- Cierra la sentencia preparada

Descripción

Estilo por procedimientos:

bool **mysqli_stmt_close** (mysqli_stmt stmt)

Estilo orientado a objetos (método):

class **mysqli_stmt** {

bool **close** (void)

}

Cierra una sentencia preparada. **mysqli_stmt_close()** también desaloja el manejador de sentencias apuntado por *stmt*. Si la sentencia actual tiene resultados pendientes o no leídos, esta función los cancela de tal manera que la siguiente consulta pueda ser ejecutada.

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Ver también

mysqli_prepare().

mysqli_stmt_data_seek

(PHP 5)

mysqli_stmt_data_seek

(no version information, might be only in CVS)

stmt->data_seek -- Busca una fila arbitrariamente en el resultado

Descripción

Estilo por procedimientos:

bool **mysqli_stmt_data_seek** (mysqli_stmt statement, int offset)

Estilo orientado a objetos (método):

class **mysqli_stmt** {

bool **data_seek** (int offset)

}

La función **mysqli_stmt_data_seek()** busca por una posición arbitraria en el resultado, especificada por *offset* en el resultado representado por *statement*. El parámetro *offset* debe ser un valor entre cero y el número total de filas menos 1 (0..**mysqli_stmt_num_rows()** - 1).

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Ver también

mysqli_prepare().

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
/* Open a connection */
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER BY Name";
if ($stmt = $mysqli->prepare($query)) {

    /* execute query */
    $stmt->execute();

    /* bind result variables */
    $stmt->bind_result($name, $code);

    /* store result */
    $stmt->store_result();

    /* seek to row no. 400 */
    $stmt->data_seek(399);

    /* fetch values */
    $stmt->fetch();

    printf ("City: %s  Countrycode: %s\n", $name, $code);

    /* close statement */
    $stmt->close();
}

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
/* Open a connection */
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER BY Name";
if ($stmt = mysqli_prepare($link, $query)) {

    /* execute query */
    mysqli_stmt_execute($stmt);

    /* bind result variables */
    mysqli_stmt_bind_result($stmt, $name, $code);

    /* store result */
    mysqli_stmt_store_result($stmt);

    /* seek to row no. 400 */
    mysqli_stmt_data_seek($stmt, 399);

    /* fetch values */
    mysqli_stmt_fetch($stmt);

    printf ("City: %s  Countrycode: %s\n", $name, $code);

    /* close statement */
    mysqli_stmt_close($stmt);
}

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
City: Benin City  Countrycode: NGA
```

mysqli_stmt_errno

(PHP 5)

mysqli_stmt_errno

(no version information, might be only in CVS)

mysqli_stmt->errno -- Regresa el código de error para la llamada más reciente

Descripción

Estilo por procedimientos:

int **mysqli_stmt_errno** (mysqli_stmt stmt)

Estilo orientado a objetos (propiedad):

class **mysqli_stmt** {

int errno

}

Para la sentencia específicada por *stmt*, **mysqli_stmt_errno()** regresa el código de error para la función más recientemente ejecutada, que pudo ser exitosa o fallar.

> **Nota:** Los números de los mensajes de error del cliente están listados en el archivo de encabezados `errmsg.h` de MySQL, Los números de los mensajes de error del servidor están listados en `mysqld_error.h`. En la distribución de los fuentes de MySQL puede encontrar una lista completa de los mensajes de error y los números de error en el archivo `Docs/mysqld_error.txt`.

Valores retornados

Un valor de código de error. Cero significa que no ocurrió error.

Ver también

mysqli_stmt_error(), y **mysqli_stmt_sqlstate()**.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
/* Open a connection */
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$mysqli->query("CREATE TABLE myCountry LIKE Country");
$mysqli->query("INSERT INTO myCountry SELECT * FROM Country");


$query = "SELECT Name, Code FROM myCountry ORDER BY Name";
if ($stmt = $mysqli->prepare($query)) {

    /* drop table */
    $mysqli->query("DROP TABLE myCountry");

    /* execute query */
    $stmt->execute();

    printf("Error: %d.\n", $stmt->errno);

    /* close statement */
    $stmt->close();
}

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
/* Open a connection */
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

mysqli_query($link, "CREATE TABLE myCountry LIKE Country");
mysqli_query($link, "INSERT INTO myCountry SELECT * FROM Country");


$query = "SELECT Name, Code FROM myCountry ORDER BY Name";
if ($stmt = mysqli_prepare($link, $query)) {

    /* drop table */
    mysqli_query($link, "DROP TABLE myCountry");

    /* execute query */
    mysqli_stmt_execute($stmt);

    printf("Error: %d.\n", mysqli_stmt_errno($stmt));

    /* close statement */
    mysqli_stmt_close($stmt);
}

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Error: 1146.
```

mysqli_stmt_error

(PHP 5)

mysqli_stmt_error

(no version information, might be only in CVS)

mysqli_stmt->error -- Regresa una descipción para el último error

Descripción

Estilo por procedimientos:

cadena **mysqli_stmt_error** (mysqli_stmt stmt)

Estilo orientado a objetos (propiedad):

class **mysqli_stmt** {

cadena error

}

Para la sentencia específicada por *stmt*, **mysqli_stmt_error()** regresa un mensaje de error para la llamada más reciente a una sentencia SQL, que pudo ser exitosa o fallar.

Valores retornados

Una cadena que describe el error. Una cadena vacía si no hubo error.

Ver también

mysqli_stmt_errno(), y **mysqli_stmt_sqlstate()**.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
/* Open a connection */
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$mysqli->query("CREATE TABLE myCountry LIKE Country");
$mysqli->query("INSERT INTO myCountry SELECT * FROM Country");


$query = "SELECT Name, Code FROM myCountry ORDER BY Name";
if ($stmt = $mysqli->prepare($query)) {

    /* drop table */
    $mysqli->query("DROP TABLE myCountry");

    /* execute query */
    $stmt->execute();

    printf("Error: %s.\n", $stmt->error);

    /* close statement */
    $stmt->close();
}

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
/* Open a connection */
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

mysqli_query($link, "CREATE TABLE myCountry LIKE Country");
mysqli_query($link, "INSERT INTO myCountry SELECT * FROM Country");


$query = "SELECT Name, Code FROM myCountry ORDER BY Name";
if ($stmt = mysqli_prepare($link, $query)) {

    /* drop table */
    mysqli_query($link, "DROP TABLE myCountry");

    /* execute query */
    mysqli_stmt_execute($stmt);

    printf("Error: %s.\n", mysqli_stmt_error($stmt));

    /* close statement */
    mysqli_stmt_close($stmt);
}

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Error: Table 'world.myCountry' doesn't exist.
```

mysqli_stmt_execute

(PHP 5)

mysqli_stmt_execute

(no version information, might be only in CVS)

stmt->execute -- Ejecuta una consulta preparada

Descripción

Estilo por procedimientos:

bool **mysqli_stmt_execute** (mysqli_stmt stmt)

Estilo orientado a objetos (método):

class **mysqli_stmt** {

bool **execute** (void)

}

La función **mysqli_stmt_execute()** ejecuta una consulta que había sido previamente preparada usando la función **mysqli_prepare()** representada por el objeto *stmt*. Cuando se ejecuta cualquier marcador de parámetro que exista será automáticamente remplazado con los datos apropiados.

Si la sentencia es UPDATE, DELETE, o INSERT, el número total de filas afectadas puede ser determinada usabdo la función **mysqli_stmt_affected_rows()**. Asimismo, si la consulta produce un resultado la función **mysqli_fetch()** es usada.

> **Nota:** Cuando se usa **mysqli_stmt_execute()**, la función **mysqli_fetch()** debe ser usada para recuperar los datos antes de realizar cualquier consulta adicional.

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Ver también

mysqli_prepare(), y **mysqli_stmt_bind_param()**.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$mysqli->query("CREATE TABLE myCity LIKE City");

/* Prepare an insert statement */
$query = "INSERT INTO myCity (Name, CountryCode, District) VALUES (?,?,?)";
$stmt = $mysqli->prepare($query);

$stmt->bind_param("sss", $val1, $val2, $val3);

$val1 = 'Stuttgart';
$val2 = 'DEU';
$val3 = 'Baden-Wuerttemberg';

/* Execute the statement */
$stmt->execute();

$val1 = 'Bordeaux';
$val2 = 'FRA';
$val3 = 'Aquitaine';

/* Execute the statement */
$stmt->execute();


 close statement */
$stmt->close();

/* retrieve all rows from myCity */
$query = "SELECT Name, CountryCode, District FROM myCity";
if ($result = $mysqli->query($query)) {
    while ($row = $result->fetch_row()) {
        printf("%s (%s,%s)\n", $row[0], $row[1], $row[2]);
    }
    /* free result set */
    $result->close();
}

/* remove table */
$mysqli->query("DROP TABLE myCity");

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

mysqli_query($link, "CREATE TABLE myCity LIKE City");

/* Prepare an insert statement */
$query = "INSERT INTO myCity (Name, CountryCode, District) VALUES (?,?,?)";
$stmt = mysqli_prepare($link, $query);

mysqli_stmt_bind_param($stmt, "sss", $val1, $val2, $val3);


$val1 = 'Stuttgart';
$val2 = 'DEU';
$val3 = 'Baden-Wuerttemberg';

/* Execute the statement */
mysqli_stmt_execute($stmt);

$val1 = 'Bordeaux';
$val2 = 'FRA';
$val3 = 'Aquitaine';

/* Execute the statement */
mysqli_stmt_execute($stmt);

/* close statement */
mysqli_stmt_close($stmt);

/* retrieve all rows from myCity */
$query = "SELECT Name, CountryCode, District FROM myCity";
if ($result = mysqli_query($link, $query)) {
    while ($row = mysqli_fetch_row($result)) {
        printf("%s (%s,%s)\n", $row[0], $row[1], $row[2]);
    }
    /* free result set */
    mysqli_free_result($result);
}

/* remove table */
mysqli_query($link, "DROP TABLE myCity");

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Stuttgart (DEU,Baden-Wuerttemberg)
Bordeaux (FRA,Aquitaine)
```

mysqli_stmt_fetch

(PHP 5)

mysqli_stmt_fetch

(no version information, might be only in CVS)

stmt->fetch -- Obtiene el resultado de una sentencia SQL preparada en las variables enlazadas

Descripción

Estilo por procedimientos:

mixto **mysqli_stmt_fetch** (mysqli_stmt stmt)

Estilo orientado a objetos (método):

class **mysqli_stmt** {

mixto **fetch** (void)

}

mysqli_stmt_fetch() obtiene el resultado de una sentencia preparada en las variables ligadas por **mysqli_stmt_bind_result()**.

> **Nota:** Note que todas las columnas deben estar enlazadas por la aplicación antes de llamar **mysqli_stmt_fetch()**.

Valores retornados

Tabla 1. Valores retornados

Valor	Descripción
TRUE	Exito. Los datos han sido obtenidos
FALSE	Ocurrió un error
NULL	No existen más filas/datos

Ver también

mysqli_prepare(), **mysqli_stmt_errno()**, **mysqli_stmt_error()**, y **mysqli_stmt_bind_result()**.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER by ID DESC LIMIT 150,5";

if ($stmt = $mysqli->prepare($query)) {

    /* execute statement */
    $stmt->execute();

    /* bind result variables */
    $stmt->bind_result($name, $code);

    /* fetch values */
    while ($stmt->fetch()) {
        printf ("%s (%s)\n", $name, $code);
    }

    /* close statement */
    $stmt->close();
}

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER by ID DESC LIMIT 150,5";

if ($stmt = mysqli_prepare($link, $query)) {

    /* execute statement */
    mysqli_stmt_execute($stmt);

    /* bind result variables */
    mysqli_stmt_bind_result($stmt, $name, $code);

    /* fetch values */
    while (mysqli_stmt_fetch($stmt)) {
        printf ("%s (%s)\n", $name, $code);
    }

    /* close statement */
    mysqli_stmt_close($stmt);
}

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Rockford (USA)
Tallahassee (USA)
Salinas (USA)
Santa Clarita (USA)
Springfield (USA)
```

mysqli_stmt_free_result

(PHP 5)

mysqli_stmt_free_result

(no version information, might be only in CVS)

stmt->free_result -- Libera la memoria donde se almacenó el resultado

Descripción

Estilo por procedimientos:

void **mysqli_stmt_free_result** (mysqli_stmt stmt)

Estilo orientado a objetos (método):

class **mysqli_stmt** {

void **free_result** (void)

}

La función **mysqli_stmt_free_result()** libera la memoria del resultado asociado con la sentencia representada por el parámetro *stmt*, el cual fue asignado por **mysqli_stmt_store_result()**.

Valores retornados

Esta función no regresa ningun valor.

Ver también

mysqli_stmt_store_result().

mysqli_stmt_init

(PHP 5)

mysqli_stmt_init

(no version information, might be only in CVS)

mysqli->stmt_init -- Inicializa una sentencia y regresa un objeto para ser usado con mysqli_stmt_prepare

Descripción

Estilo por procedimientos :

mysqli_stmt **mysqli_stmt_init** (mysqli identificador_de_enlace)

Estilo orientado a objetos (propiedad):

class **mysqli** {

mysqli_stmt **stmt_init** (void)

}

Asigna e inicializa una objeto de sentencia conveniente para **mysqli_stmt_prepare()**.

> **Nota:** Cualquier llamada subsecuente a cualquier función mysqli_stmt fallará hasta que sea llamada **mysqli_stmt_prepare()**.

Valores retornados

Regresa un objeto.

Ver también

mysqli_stmt_prepare().

mysqli_stmt_num_rows

(PHP 5)

mysqli_stmt_num_rows

(no version information, might be only in CVS)

stmt->num_rows -- Regresa el número de filas en un resultado de la sentencia

Descripción

Estilo por procedimientos :

mixto **mysqli_stmt_num_rows** (mysqli_stmt stmt)

Estilo orientado a objetos (propiedad):

class **mysqli_stmt** {

int num_rows

}

Regresa el número de filas en el resultado. El uso de **mysqli_stmt_num_rows()** depende de que use **mysqli_stmt_store_result()** para almacenar todo el resultado en el manejador de la sentencia.

Si usa **mysqli_stmt_store_result()**, **mysqli_stmt_num_rows()** debe ser llamada inmediatamente.

Valores retornados

Un entero representando el número de filas en el resultado.

Ver también

mysqli_stmt_affected_rows(), **mysqli_prepare()**, y **mysqli_stmt_store_result()**.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
/* Open a connection */
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER BY Name LIMIT 20";
if ($stmt = $mysqli->prepare($query)) {

    /* execute query */
    $stmt->execute();

    /* store result */
    $stmt->store_result();

    printf("Number of rows: %d.\n", $stmt->num_rows);

    /* close statement */
    $stmt->close();
}

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
/* Open a connection */
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER BY Name LIMIT 20";
if ($stmt = mysqli_prepare($link, $query)) {

    /* execute query */
    mysqli_stmt_execute($stmt);

    /* store result */
    mysqli_stmt_store_result($stmt);

    printf("Number of rows: %d.\n", mysqli_stmt_num_rows($stmt));

    /* close statement */
    mysqli_stmt_close($stmt);
}

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Number of rows: 20.
```

mysqli_stmt_param_count

(PHP 5)

mysqli_stmt_param_count

(no version information, might be only in CVS)

stmt->param_count -- Regresa el número de parámetros para la sentencia dada

Descripción

Estilo por procedimientos:

int **mysqli_stmt_param_count** (mysqli_stmt stmt)

Estilo orientado a objetos (propiedad):

class **mysqli_stmt** {

int param_count

}

mysqli_stmt_param_count() regresa el número de marcadores de parámetro presentes en la sentencia preparada.

Valores retornados

Regresa un entero representando el número de parámetros.

Ver también

mysqli_prepare().

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

if ($stmt = $mysqli->prepare("SELECT Name FROM Country WHERE Name=? OR Code=?")) {

    $marker = $stmt->param_count;
    printf("Statement has %d markers.\n", $marker);

    /* close statement */
    $stmt->close();
}

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

if ($stmt = mysqli_prepare($link, "SELECT Name FROM Country WHERE Name=? OR Code=?")) {

    $marker = mysqli_stmt_param_count($stmt);
    printf("Statement has %d markers.\n", $marker);

    /* close statement */
    mysqli_stmt_close($stmt);
}

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Statement has 2 markers.
```

mysqli_stmt_prepare

(PHP 5)

mysqli_stmt_prepare

(no version information, might be only in CVS)

stmt->prepare -- Prepara una sentencia SQL para su ejecución

Descripción

Estilo por procedimientos:

bool **mysqli_stmt_prepare** (mysqli_stmt stmt, cadena query)

Estilo orientado a objetos (método)

class **mysqli_stmtt** {

mixto **prepare** (cadena query)

}

mysqli_stmt_prepare() prepara la consulta SQL apuntada por la cadena de consulta terminada en NULL. El objeto de la sentencia tiene que ser asignado por **mysqli_stmt_init()**. La consulta debe consistir de una sentencia SQL sencilla.

> **Nota:** No debe agregar al final el punto y coma o \g a la sentencia.

El parámetro *query* puede incluir uno o más marcadores de parámetro en la sentencia SQL, insertando el caracter interrogativo (*?*) en la posición apropiada.

> **Nota:** Los marcadores son legales solo en ciertos lugares en la sentencia SQL. Por ejemplo, son permitidos en la lista VALUES() de una sentencia INSERT (para específicar calores de columnas para una fila), o en una comparación con una columna en una clausula WHERE para específicar un valor de comparación.
>
> Sin embargo, no son permitidos para identificadores (tales como nombres de tabla o columna), en el listado de los nombres de las columnas a ser regresadas por la sentencia SELECT, o para específicar operadores de resultado binario, tales como = el signo de igual. La última restricción es necesaria porque serí imposuble determinar el tipo de parámetro. En general, los marcadores de parámetro son legales solo en las sentencias del lenguaje de manipulación de datos (DML), y no en las sentencias del lenguaje de definición de datos (DDL).

Los marcadores de parámetro deben estar enlazados a variables de aplicación usando **mysqli_stmt_bind_param()** y/o **mysqli_stmt_bind_result()** antes de ejecutar la sentencia u obtener las filas del resultado.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ver también

mysqli_stmt_init(), **mysqli_stmt_execute()**, **mysqli_stmt_fetch()**, **mysqli_stmt_bind_param()**, **mysqli_stmt_bind_result()**, y **mysqli_stmt_close()**.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$city = "Amersfoort";

/* create a prepared statement */
$stmt =  $mysqli->stmt_init();
if ($stmt->prepare("SELECT District FROM City WHERE Name=?")) {

    /* bind parameters for markers */
    $stmt->bind_param("s", $city);

    /* execute query */
    $stmt->execute();

    /* bind result variables */
    $stmt->bind_result($district);

    /* fetch value */
    $stmt->fetch();

    printf("%s is in district %s\n", $city, $district);

    /* close statement */
    $stmt->close();
}

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$city = "Amersfoort";

/* create a prepared statement */
$stmt = mysqli_stmt_init();
if ($stmt = mysqli_stmt_prepare($stmt, "SELECT District FROM City WHERE Name=?")) {

    /* bind parameters for markers */
    mysqli_stmt_bind_param($stmt, "s", $city);

    /* execute query */
    mysqli_stmt_execute($stmt);

    /* bind result variables */
    mysqli_stmt_bind_result($stmt, $district);

    /* fetch value */
    mysqli_stmt_fetch($stmt);

    printf("%s is in district %s\n", $city, $district);

    /* close statement */
    mysqli_stmt_close($stmt);
}

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Amersfoort is in district Utrecht
```

mysqli_stmt_reset

(PHP 5)

mysqli_stmt_reset

(no version information, might be only in CVS)

stmt->reset -- Resetea una sentencia preparada

Descripción

Estilo por procedimientos:

bool **mysqli_stmt_reset** (mysqli_stmt stmt)

Estilo orientado a objetos (método):

class **mysqli_stmt** {

bool **reset** (void)

}

La función **mysqli_stmt_reset()** resetea una sentencia preparada en el cliente y el servidor al estado que estaba después de PREPARE. Por ahora esta es usada principalmente para resetear los datos enviados con **mysqli_stmt_send_long_data()**.

> **Nota:** Para preparar una sentencia con otra consulta SQL use la función **mysqli_stmt_prepare()**.

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Ver también

mysqli_prepare().

mysqli_stmt_result_metadata

(PHP 5)

mysqli_stmt_result_metadata -- Regresa metadatos del resultado de una sentencia preparada

Descripción

Estilo por procedimientos:

mixto **mysqli_stmt_result_metadata** (mysqli_stmt stmt)

Estilo orientado a objetos (método):

class **mysqli_stmt** {

mixto **result_metadata** (void)

}

Si una sentencia pasada a **mysqli_prepare()** es una que produce un resultado, **mysqli_stmt_result_metadata()** regresa el objeto del resultado que puede ser usado para procesar los metadatos de información tales como el número total de campos e información del campo individual.

> **Nota:** Este apuntador del resultado puede ser pasado como un argumento a cualquiera de las funciones basadas en campos que procesan los metadatos del resultado tales como:
>
> - **mysqli_num_fields()**
> - **mysqli_fetch_field()**

- **mysqli_fetch_field_direct()**
- **mysqli_fetch_fields()**
- **mysqli_field_count()**
- **mysqli_field_seek()**
- **mysqli_field_tell()**
- **mysqli_free_result()**

La estructura del resultado debe ser liberada cuando haya terminado con esta, lo cual puede hacer pasandola a **mysqli_free_result()**

> **Nota:** El resultado regresado por **mysqli_stmt_result_metadata()** contiene solo metadatos. No contiene ninguna fila del resultado. Las filas son obtenidas usando el manejador de la sentencia con **mysqli_fetch()**.

Valores retornados

mysqli_stmt_result_metadata() regresa un objeto del resultado o **`FALSE`** si ocurrio un error.

Ver también

mysqli_prepare(), y **mysqli_free_result()**.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "test");

$mysqli->query("DROP TABLE IF EXISTS friends");
$mysqli->query("CREATE TABLE friends (id int, name varchar(20))");

$mysqli->query("INSERT INTO friends VALUES (1,'Hartmut'), (2, 'Ulf')");

$stmt = $mysqli->prepare("SELECT id, name FROM friends");
$stmt->execute();

/* get resultset for metadata */
$result = $stmt->result_metadata();

/* retrieve field information from metadata result set */
$field = $result->fetch_field();

printf("Fieldname: %s\n", $field->name);

/* close resultset */
$result->close();

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```php
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "test");

mysqli_query($link, "DROP TABLE IF EXISTS friends");
mysqli_query($link, "CREATE TABLE friends (id int, name varchar(20))");

mysqli_query($link, "INSERT INTO friends VALUES (1,'Hartmut'), (2, 'Ulf')");

$stmt = mysqli_prepare($link, "SELECT id, name FROM friends");
mysqli_stmt_execute($stmt);

/* get resultset for metadata */
$result = mysqli_stmt_result_metadata($stmt);

/* retrieve field information from metadata result set */
$field = mysqli_fetch_field($result);

printf("Fieldname: %s\n", $field->name);

/* close resultset */
mysqli_free_result($result);

/* close connection */
mysqli_close($link);
?>
```

mysqli_stmt_send_long_data

(PHP 5)

mysqli_stmt_send_long_data

(no version information, might be only in CVS)

stmt->send_long_data -- Envía los datos en bloques

Descripción

Estilo por procedimientos:

bool **mysqli_stmt_send_long_data** (mysqli_stmt stmt, int param_nr, cadena data)

Estilo orientado a objetos (método)

class **mysqli_stmt** {

bool **send_long_data** (int param_nr, cadena data)

}

Permite enviar datos de parámetro al servidor en piezas (o pedazos) ej. si el tamaño de un dato BLOB excede el tamaño de *max_allowed_packet*. Esta función puede ser llamada multiples veces para enviar las partes de un caracter o valor de datos binario de una columna, la cual debe ser uno de los tipos de datos TEXT o BLOB.

param_nr indica cual parámetro asociar con los datos. Los parámetros son numerados empezando desde cero. *data* es una cadena conteniendo los datos a ser enviados.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ver también

mysqli_prepare(), y **mysqli_stmt_bind_param()**.

mysqli_stmt_sqlstate

(PHP 5)

mysqli_stmt_sqlstate -- Regresa el error SQLSTATE de la operación con sentencias previa

Descripción

cadena **mysqli_stmt_sqlstate** (mysqli_stmt stmt)

Regresa una cadena conteniendo el código de error SQLSTATE para la sentencia preparada más recientemente invocada que pudo ser exitosa o fallar. El código de error consiste de cinco caracteres. *'00000'* significa que no hubo error. Los valores son específicados por ANSI SQL y ODBC. Para una lista de los posibles valores vea http://dev.mysql.com/doc/mysql/en/Error-returns.html.

> **Nota:** Note que no todos los errores de MySQL tienen un correspondiente en SQLSTATE. El valor *HY000* (error general) es usado para errores que no tienen ese correspondiente.

Valores retornados

Regresa una candena conteniendo el código de error SQLSTATE para el último error. El código de error consiste de cinco caracteres. *'00000'* significa que no hubo error.

Ver también

mysqli_stmt_errno(), y **mysqli_stmt_error()**.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```php
<?php
/* Open a connection */
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$mysqli->query("CREATE TABLE myCountry LIKE Country");
$mysqli->query("INSERT INTO myCountry SELECT * FROM Country");


$query = "SELECT Name, Code FROM myCountry ORDER BY Name";
if ($stmt = $mysqli->prepare($query)) {

    /* drop table */
    $mysqli->query("DROP TABLE myCountry");

    /* execute query */
    $stmt->execute();

    printf("Error: %s.\n", $stmt->sqlstate);

    /* close statement */
    $stmt->close();
}

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```php
<?php
/* Open a connection */
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

mysqli_query($link, "CREATE TABLE myCountry LIKE Country");
mysqli_query($link, "INSERT INTO myCountry SELECT * FROM Country");


$query = "SELECT Name, Code FROM myCountry ORDER BY Name";
if ($stmt = mysqli_prepare($link, $query)) {

    /* drop table */
    mysqli_query($link, "DROP TABLE myCountry");

    /* execute query */
    mysqli_stmt_execute($stmt);

    printf("Error: %s.\n", mysqli_stmt_sqlstate($stmt));

    /* close statement */
    mysqli_stmt_close($stmt);
}

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Error: 42S02.
```

mysqli_stmt_store_result

(PHP 5)

mysqli_stmt_store_result

(no version information, might be only in CVS)

mysqli_stmt->store_result -- Transfiere un resultado de una sentencia preparada

Descripción

Estilo por procedimientos:

bool **mysqli_stmt_store_result** (mysqli_stmt stmt)

Estilo orientado a objetos (método):

class **mysqli_stmt** {

mysqli_result **store_result** (void)

}

Se debe llamar **mysqli_stmt_store_result()** para cada consulta que exitosamente produce un resultado (*SELECT, SHOW, DESCRIBE, EXPLAIN*), y solo si quiere almacenar el resultado completo en el lado del cliente, asi que las subsecuentes llamadas **mysqli_fetch()** regresarán los datos almacenados en la memoria intermedia (BUFFER).

> **Nota:** Es inecesario llamar **mysqli_stmt_store_result()** para otras consultas, pero si lo hace, no dañará o causará cualquier cambio en el desempeño en cualquier caso. Puede detectar si la consulta produjo un resultado checando si **mysqli_stmt_result_metadata()** regresa NULL.

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Ver también

mysqli_prepare(), **mysqli_stmt_result_metadata()**, y **mysqli_fetch()**.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```php
<?php
/* Open a connection */
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER BY Name LIMIT 20";
if ($stmt = $mysqli->prepare($query)) {

    /* execute query */
    $stmt->execute();

    /* store result */
    $stmt->store_result();

    printf("Number of rows: %d.\n", $stmt->num_rows);

    /* free result */
    $stmt->free_result();

    /* close statement */
    $stmt->close();
}

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```php
<?php
/* Open a connection */
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER BY Name LIMIT 20";
if ($stmt = mysqli_prepare($link, $query)) {

    /* execute query */
    mysqli_stmt_execute($stmt);

    /* store result */
    mysqli_stmt_store_result($stmt);

    printf("Number of rows: %d.\n", mysqli_stmt_num_rows($stmt));

    /* free result */
    mysqli_stmt_free_result($stmt);

    /* close statement */
    mysqli_stmt_close($stmt);
}

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Number of rows: 20.
```

mysqli_store_result

(PHP 5)

mysqli_store_result

(no version information, might be only in CVS)

mysqli->store_result -- Transfiere un resultado de la última consulta

Descripción

Estilo por procedimientos:

objeto **mysqli_store_result** (mysqli_stmt identificador_de_enlace)

Estilo orientado a objetos (método):

class **mysqli** {

objeto **store_result** (void)

}

Transfiere el resultado de la última consulta en la conexión de base de datos representada por el parámetro *identificador_de_enlace*, para ser usado con la función **mysqli_data_seek()**.

> **Nota:** Aunque es siempre una buena practica liberar la memoria usada por el resultado de una consulta usando la función **mysqli_free_result()**, cuando se transfieren resultados de gran tamaño usando **mysqli_store_result()** en este caso esta practica se convierte particularmente importante.
>
> **Nota: mysqli_store_result()** regresa **`FALSE`** en caso de que la consulta no regrese un resultado (si la consulta fue, por ejemplo una sentencia INSERT). Esta función también regresa **`FALSE`** si falla la lectura del resultado. Puede checar si obtuvo un error checando si **mysqli_error()** no regresa una cadena vacía, si **mysqli_errno()** regresa un valor diferente de cero o si **mysqli_field_count()** regresa un valor diferente de cero. También es una razón posible para que esta función regrese **`FALSE`** después que una llamada exitosa de **mysqli_query()** pueda ser demasiado grande el resultado (la memoria no lo puede contener). Si **mysqli_field_count()** regresa un valor no cero, la sentencia debió producir un resultado no vacío.

Valores retornados

Regresa un objeto resultado con almacenamiento intermedio (buffered) o **`FALSE`** si ocurre un error.

See also

mysqli_real_query(), y **mysqli_use_result()**.

Ejemplos

Vea **mysqli_multi_query()**.

mysqli_thread_id

(PHP 5)

mysqli_thread_id

(no version information, might be only in CVS)

mysqli->thread_id -- Regresa el identificador del THREAD para la conexión actual

Descripción

Estilo por procedimientos:

int **mysqli_thread_id** (mysqli identificador_de_enlace)

Estilo orientado a objetos (propiedad):

class **mysqli** {

int thread_id

}

La finción **mysqli_thread_id()** regresa el identificador del THREAD para la conexión actual la cual puede ser entonces destruir usando la función **mysqli_kill()**. Si la conexión se pierde y se reconecta con **mysqli_ping()**, el identificador del THREAD será otro. Por lo tanto debe obtener el identificador del THREAD solo cuando lo necesite.

> **Nota:** El identificador del THREAD es asignado en base a conexión. Por lo tanto si la conexión está rota y entonces se reestablece se le asignará un nuevo identificador de THREAD.
>
> Para destruir una consulta que se está ejecutando use el comando SQL *KILL QUERY processid*.

Valores retornados

mysqli_thread_id() regresa el identificador del THREAD para la conexión actual.

Ver también

mysqli_kill().

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

/* determine our thread id */
$thread_id = $mysqli->thread_id;

/* Kill connection */
$mysqli->kill($thread_id);

/* This should produce an error */
if (!$mysqli->query("CREATE TABLE myCity LIKE City")) {
    printf("Error: %s\n", $mysqli->error);
    exit;
}

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

/* determine our thread id */
$thread_id = mysqli_thread_id($link);

/* Kill connection */
mysqli_kill($link, $thread_id);

/* This should produce an error */
if (!mysqli_query($link, "CREATE TABLE myCity LIKE City")) {
    printf("Error: %s\n", mysqli_error($link));
    exit;
}

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Error: MySQL server has gone away
```

mysqli_thread_safe

(PHP 5)

mysqli_thread_safe -- Regresa si se ha dado un THREAD seguro o no Returns whether thread safety is given or not

Descripción

Estilo por procedimientos:

bool **mysqli_thread_safe** (void)

mysqli_thread_safe() indica si la libreria del cliente ha sido compilada como thread seguro.

Valores retornados

`TRUE` si la libreria del client es thread seguro, en otro caso **`FALSE`**.

mysqli_use_result

(PHP 5)

mysqli_use_result

(no version information, might be only in CVS)

mysqli->use_result -- Inicia una recuperación de un resultado

Descripción

Estilo por procedimientos:

mixto **mysqli_use_result** (mysqli identificador_de_enlace)

Estilo orientado a objetos (método):

class **mysqli** {

mixto **use_result** (void)

}

mysqli_use_result() es usada para iniciar la recuperación de un resultado de la última consulta ejecutada usando la función **mysqli_real_query()** en la conexión de base de datos específicda por el parámetro *identificador_de_enlace*. Esta o la funció **mysqli_store_result()** debe ser llamada antes de que el resultado de una consulta pueda ser recuperado, y una o la otra debe esr llamada para prevenir que falle la siguiente consulta en la conexión a la base de datos.

> **Nota:** La función **mysqli_use_result()** no transfiere el resultado completo de la base de datos y por lo tanto no puede ser usadas funciones tales como **mysqli_data_seek()** para moverse a una fila indicada en el resultado. Para usar esta funcionalidad, el resultado debe ser almacenado usando **mysqli_store_result()**. No se debe usar **mysqli_use_result ()** si se está haciendo demasiado procesamiento en el lado del cliente, puesto que esto sobrecargará el servidor y evitará que otros procesos puedan actualizar cualquier tabla de entre las cuales los datos están siendo obtenidos.

Valores retornados

Regresa el objeto resultante sin almacenamiento intermedio o **`FALSE`** si ha ocurrido un error.

Ver también

mysqli_real_query(), y **mysqli_store_result()**.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$query  = "SELECT CURRENT_USER();";
$query .= "SELECT Name FROM City ORDER BY ID LIMIT 20, 5";

/* execute multi query */
if ($mysqli->multi_query($query)) {
    do {
        /* store first result set */
        if ($result = $mysqli->use_result()) {
            while ($row = $result->fetch_row()) {
                printf("%s\n", $row[0]);
            }
            $result->close();
        }
        /* print divider */
        if ($mysqli->more_results()) {
            printf("-----------------\n");
        }
    } while ($mysqli->next_result());
}

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$query  = "SELECT CURRENT_USER();";
$query .= "SELECT Name FROM City ORDER BY ID LIMIT 20, 5";

/* execute multi query */
if (mysqli_multi_query($link, $query)) {
    do {
        /* store first result set */
        if ($result = mysqli_use_result($link)) {
            while ($row = mysqli_fetch_row($result)) {
                printf("%s\n", $row[0]);
            }
            mysqli_free_result($result);
        }
        /* print divider */
        if (mysqli_more_results($link)) {
            printf("-----------------\n");
        }
    } while (mysqli_next_result($link));
}

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
my_user@localhost
-----------------
Amersfoort
Maastricht
Dordrecht
Leiden
Haarlemmermeer
```

mysqli_warning_count

(PHP 5)

mysqli_warning_count

(no version information, might be only in CVS)

mysqli->warning_count -- Regresa el número de alertas de la última consulta para el identificador de enlace dado

Descripción

Estilo por procedimientos:

int **mysqli_warning_count** (mysqli identificador_de_enlace)

Estilo orientado a objetos (propiedad):

class **mysqli** {

int warning_count

}

mysqli_warning_count() regresa el número de alertas de la última consulta en la conexión representada por el parámetro *identificador_de_enlace*.

> **Nota:** Para recuperar los mensajes de alerta, puede usar el comando SQL *SHOW WARNINGS [limit row_count]*.

Valores retornados

El Número de alertas o cero si no hay alertas.

Ver también

mysqli_errno(), **mysqli_error()**, y **mysqli_sqlstate()**.

Ejemplos

Ejemplo 1. Estilo orientado a objetos

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

$mysqli->query("CREATE TABLE myCity LIKE City");

/* a remarkable city in Wales */
$query = "INSERT INTO myCity (CountryCode, Name) VALUES('GBR',
        'Llanfairpwllgwyngyllgogerychwyrndrobwllllantysiliogogogoch')";

$mysqli->query($query);

if ($mysqli->warning_count) {
    if ($result = $mysqli->query("SHOW WARNINGS")) {
        $row = $result->fetch_row();
        printf("%s (%d): %s\n", $row[0], $row[1], $row[2]);
        $result->close();
    }
}

/* close connection */
$mysqli->close();
?>
```

Ejemplo 2. Estilo por procedimientos

```php
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

mysqli_query($link, "CREATE TABLE myCity LIKE City");

/* a remarkable long city name in Wales */
$query = "INSERT INTO myCity (CountryCode, Name) VALUES('GBR',
        'Llanfairpwllgwyngyllgogerychwyrndrobwllllantysiliogogogoch')";

mysqli_query($link, $query);

if (mysqli_warning_count($link)) {
    if ($result = mysqli_query($link, "SHOW WARNINGS")) {
        $row = mysqli_fetch_row($result);
        printf("%s (%d): %s\n", $row[0], $row[1], $row[2]);
        mysqli_free_result($result);
    }
}

/* close connection */
mysqli_close($link);
?>
```

El resultado del ejemplo seria:

```
Warning (1264): Data truncated for column 'Name' at row 1
```

LXXXI. Funciones de Control de Pantalla con Terminal Ncurses

Introducción

ncurses (new curses) es un sistema de emulación del paquete curses del Sistema V 4.0 (y superiores). Usa formatos terminfo, soporta pads, colores, resaltados múltiples, caracteres de formulario y asignaciones de funciones de teclado. Debido a la naturaleza interactiva de esta biblioteca, ésta será de poca utilidad para la escritura de aplicaciones Web, pero puede ser útil cuando se escriben scripts orientados al uso de PHP desde la línea de comandos.

Aviso
Esta extensión es *EXPERIMENTAL*. Esto significa que el comportamiento de esta extensión, los nombre de sus funciones y en definitiva TODO lo documentado sobre esta extensión, puede cambiar en una futura versión de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Ncurses se encuentra disponible para las siguientes plataformas:

- AIX
- BeOS
- Cygwin
- Digital Unix (aka OSF1)

- FreeBSD
- GNU/Linux
- HPUX
- IRIX
- OS/2
- SCO OpenServer
- Solaris
- SunOS

Requirimientos

Necesita las bibliotecas ncurses y sus archivos de cabecera. Descargue la última versión desde ftp://ftp.gnu.org/pub/gnu/ncurses/ o algún otro mirror GNU.

Instalación

Para que estas funciones trabajen, debe compilar la versión CGI o CLI de PHP con *--with-ncurses [=DIR]*.

Configuración en tiempo de ejecución

El comportamiento de estas funciones está afectado por los valores definidos en `php.ini`.

Tabla 1. Opciones de configuración de ncurses

Nombre	Predeterminado	Modificable
ncurses.value	"42"	PHP_INI_ALL
ncurses.string	"foobar"	PHP_INI_ALL

Para más detalles sobre las constantes PHP_INI_* y su definición, vea **ini_set()**.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

Códigos de error

En caso de fallos, las funciones ncurses devuelven NCURSES_ERR.

Colores

Tabla 2. constantes de color de ncurses

constante	significado
NCURSES_COLOR_BLACK	sin color (negro)
NCURSES_COLOR_WHITE	blanco
NCURSES_COLOR_RED	rojo - soportado cuando la terminal se encuentra en modo de color
NCURSES_COLOR_GREEN	verde - soportado cuando la terminal se encuentra en modo de color
NCURSES_COLOR_YELLOW	amarillo - soportado cuando la terminal se encuentra en modo de color
NCURSES_COLOR_BLUE	azul - soportado cuando la terminal se encuentra en modo de color
NCURSES_COLOR_CYAN	cyan - soportado cuando la terminal se encuentra en modo de color
NCURSES_COLOR_MAGENTA	magenta - soportado cuando la terminal se encuentra en modo de color

---

Teclas

Tabla 3. constantes de teclas ncurses

constante	significado
NCURSES_KEY_F0 - NCURSES_KEY_F64	teclas de función F1 - F64
NCURSES_KEY_DOWN	flecha hacia abajo
NCURSES_KEY_UP	flecha hacia arriba
NCURSES_KEY_LEFT	flecha hacia la izquierda
NCURSES_KEY_RIGHT	flecha hacia la derecha
NCURSES_KEY_HOME	tecla home (flecha arriba+izquierda)
NCURSES_KEY_BACKSPACE	backspace

constante	significado
NCURSES_KEY_DL	eliminar línea
NCURSES_KEY_IL	insertar línea
NCURSES_KEY_DC	eliminar caracter
NCURSES_KEY_IC	insertar caracter o entrar en modo de inserción
NCURSES_KEY_EIC	salir de modo de inserción de caracteres
NCURSES_KEY_CLEAR	limpiar la pantalla
NCURSES_KEY_EOS	limpiar hasta el fin de la pantalla
NCURSES_KEY_EOL	limpiar hasta el fin de la línea
NCURSES_KEY_SF	desplazarse una línea hacia adelante
NCURSES_KEY_SR	desplazarse una línea hacia atrás
NCURSES_KEY_NPAGE	siguiente página
NCURSES_KEY_PPAGE	página anterior
NCURSES_KEY_STAB	definir tab
NCURSES_KEY_CTAB	eliminar tab
NCURSES_KEY_CATAB	eliminar todos los tabs
NCURSES_KEY_SRESET	reset suave (parcial)
NCURSES_KEY_RESET	reset o reset fuerte
NCURSES_KEY_PRINT	imprimir
NCURSES_KEY_LL	izquierda inferior
NCURSES_KEY_A1	izquierda superior del teclado numérico
NCURSES_KEY_A3	derecha superior del teclado numérico
NCURSES_KEY_B2	centro del teclado numérico
NCURSES_KEY_C1	izquierda inferior del teclado numérico
NCURSES_KEY_C3	derecha inferior del teclado numérico
NCURSES_KEY_BTAB	tab hacia atrás
NCURSES_KEY_BEG	comienzo
NCURSES_KEY_CANCEL	cancelar
NCURSES_KEY_CLOSE	cerrar
NCURSES_KEY_COMMAND	cmd (comando)
NCURSES_KEY_COPY	copiar
NCURSES_KEY_CREATE	crear
NCURSES_KEY_END	fin
NCURSES_KEY_EXIT	salida
NCURSES_KEY_FIND	encontrar
NCURSES_KEY_HELP	ayuda
NCURSES_KEY_MARK	marca
NCURSES_KEY_MESSAGE	mensaje

constante	significado
NCURSES_KEY_MOVE	mover
NCURSES_KEY_NEXT	siguiente
NCURSES_KEY_OPEN	abrir
NCURSES_KEY_OPTIONS	opciones
NCURSES_KEY_PREVIOUS	anterior
NCURSES_KEY_REDO	rehacer
NCURSES_KEY_REFERENCE	ref (referencia)
NCURSES_KEY_REFRESH	refrescar
NCURSES_KEY_REPLACE	reemplazar
NCURSES_KEY_RESTART	reiniciar
NCURSES_KEY_RESUME	reiniciar
NCURSES_KEY_SAVE	guardar
NCURSES_KEY_SBEG	comienzo usando shift
NCURSES_KEY_SCANCEL	cancelar usando shift
NCURSES_KEY_SCOMMAND	comando usando shift
NCURSES_KEY_SCOPY	copiar usando shift
NCURSES_KEY_SCREATE	crear usando shift
NCURSES_KEY_SDC	eliminar caracter usando shift
NCURSES_KEY_SDL	eliminar línea usando shift
NCURSES_KEY_SELECT	seleccionar
NCURSES_KEY_SEND	final usando shift
NCURSES_KEY_SEOL	fin de línea usando shift
NCURSES_KEY_SEXIT	salida usando shift
NCURSES_KEY_SFIND	encontrar usando shift
NCURSES_KEY_SHELP	ayuda usando shift
NCURSES_KEY_SHOME	home usando shift
NCURSES_KEY_SIC	entrada usando shift
NCURSES_KEY_SLEFT	flecha hacia la izquierda usando shift
NCURSES_KEY_SMESSAGE	mensaje usando shift
NCURSES_KEY_SMOVE	mover usando shift
NCURSES_KEY_SNEXT	siguiente usando shift
NCURSES_KEY_SOPTIONS	opciones usando shift
NCURSES_KEY_SPREVIOUS	anterior usando shift
NCURSES_KEY_SPRINT	imprimir usando shift
NCURSES_KEY_SREDO	rehacer usando shift
NCURSES_KEY_SREPLACE	reemplazar usando shift
NCURSES_KEY_SRIGHT	flecha hacia la derecha usando shift
NCURSES_KEY_SRSUME	reiniciar usando shift

constante	significado
NCURSES_KEY_SSAVE	guardar usando shift
NCURSES_KEY_SSUSPEND	suspender usando shift
NCURSES_KEY_UNDO	deshacer
NCURSES_KEY_MOUSE	evento del mouse ha ocurrido
NCURSES_KEY_MAX	valor máximo de tecla

Mouse

Tabla 4. constantes de mouse

Constante	significado
NCURSES_BUTTON1_RELEASED - NCURSES_BUTTON4_RELEASED	botón (1-4) liberado
NCURSES_BUTTON1_PRESSED - NCURSES_BUTTON4_PRESSED	botón (1-4) presionado
NCURSES_BUTTON1_CLICKED - NCURSES_BUTTON4_CLICKED	botón (1-4) pulsado
NCURSES_BUTTON1_DOUBLE_CLICKED - NCURSES_BUTTON4_DOUBLE_CLICKED	botón (1-4) pulsado dos veces
NCURSES_BUTTON1_TRIPLE_CLICKED - NCURSES_BUTTON4_TRIPLE_CLICKED	botón (1-4) pulsado tres veces
NCURSES_BUTTON_CTRL	ctrl presionado durante el clic
NCURSES_BUTTON_SHIFT	shift presionado durante el clic
NCURSES_BUTTON_ALT	alt presionado durante el clic
NCURSES_ALL_MOUSE_EVENTS	reportar todos los eventos del mouse
NCURSES_REPORT_MOUSE_POSITION	reportar la posición del mouse

Tabla de contenidos

ncurses_bottom_panel -- Mueve un panel visible al fondo de la pila
ncurses_can_change_color -- Chequear si es posible cambiar los colores de la terminal
ncurses_cbreak -- Cambio de búferes de entrada
ncurses_clear -- Limpiar la pantalla
ncurses_clrtobot -- Limpiar la pantalla desde la posición actual al final
ncurses_clrtoeol -- Limpiar la pantalla desde la posición actual al final de la línea
ncurses_color_content -- Obtiene el valor RGB del color
ncurses_color_set -- Establecer los colores de primer y segundo plano
ncurses_curs_set -- Establecer el estado del cursor
ncurses_def_prog_mode -- Guarda los modos de terminal (programa)
ncurses_def_shell_mode -- Guarda los modos de terminal (intérprete de comandos)
ncurses_define_key -- Definir un código de tecla
ncurses_del_panel -- Remover un panel de la pila y eliminarlo (pero no la ventana asociada)
ncurses_delay_output -- Retrasar la salida en la terminal usando caracteres de relleno
ncurses_delch -- Eliminar un caracter en la posición actual, mover el resto de la línea hacia la izquierda
ncurses_deleteln -- Eliminar una línea en la posición actual, mover el resto de la pantalla hacia arriba
ncurses_delwin -- Eliminar una ventana ncurses
ncurses_doupdate -- Escribir todas las actualizaciones preparadas sobre la terminal
ncurses_echo -- Activar la repetición de entrada del teclado
ncurses_echochar -- Salida de un caracter sencillo, incluyendo actualización
ncurses_end -- Dejar de usar ncurses, limpiar la pantalla
ncurses_erase -- Limpiar pantalla de terminal
ncurses_erasechar -- Devuelve el caracter de borrado actual
ncurses_filter -- Definir LINES para iniscr() y newterm() a 1
ncurses_flash -- Relampaguear la pantalla de terminal (campana visual)
ncurses_flushinp -- Volcar el búfer de entrada de teclado
ncurses_getch -- Leer un caracter desde el teclado
ncurses_getmaxyx -- Devuelve el tamaño de una ventana
ncurses_getmouse -- Lee un evento del mouse
ncurses_getyx -- Devuelve la posición de cursor actual para una ventana
ncurses_halfdelay -- Colocar la terminal en modo de medio retraso
ncurses_has_colors -- Chequear si la terminal tiene colores
ncurses_has_ic -- Chequear por el soporte de inserción y borrado
ncurses_has_il -- Chequea el soporte de inserción y borrado
ncurses_has_key -- Chequear por la presencia de una tecla de función en el teclado de la terminal
ncurses_hide_panel -- Remover un panel de la pila, haciéndolo invisible
ncurses_hline -- Dibujar una línea horizontal en la posición actual usando un caracter con atributos, y de máximo n caracteres de largo
ncurses_inch -- Obtener el caracter y los atributos de la posición actual
ncurses_init_color -- Establecer un nuevo valor RGB para un color
ncurses_init_pair -- Reservar una pareja de color
ncurses_init -- Inicializar ncurses
ncurses_insch -- Insertar un caracter, moviendo el resto de la línea, incluyendo el caracter en la posición actual
ncurses_insdelln -- Insertar líneas antes de la línea actual, desplazando hacia abajo (los números negativos eliminan líneas y desplazan hacia arriba)
ncurses_insertln -- Insertar una línea, mover el resto de la pantalla hacia abajo
ncurses_insstr -- Insertar cadena en la posición actual, moviendo el resto de la línea hacia la derecha
ncurses_instr -- Lee una cadena desde la pantalla de terminal
ncurses_isendwin -- Ncurses se encuentra en modo endwin, puede efectuarse salida de pantalla normal
ncurses_keyok -- Habilitar o deshabilitar un código de tecla
ncurses_keypad -- Activa o desactiva el teclado numérico

ncurses_killchar -- Devuelve el caracter de eliminación de línea actual
ncurses_longname -- Devuelve la descripción de la terminal
ncurses_meta -- Habilita/deshabilita la meta-información de 8-bits de tecla
ncurses_mouse_trafo -- Transforma coordenadas
ncurses_mouseinterval -- Establecer el tiempo de espera para clics de botón del mouse
ncurses_mousemask -- Establece opciones del mouse
ncurses_move_panel -- Mueve un panel de modo que su esquina superior izquierda se encuentre en [comienzo_x, comienzo_y]
ncurses_move -- Mover la posición de salida
ncurses_mvaddch -- Mover la posición actual y agregar un caracter
ncurses_mvaddchnstr -- Mover la posición y agregar una cadena con atributos con la longitud especificada
ncurses_mvaddchstr -- Mover la posición y agregar una cadena con atributos
ncurses_mvaddnstr -- Mover la posición y agregar una cadena con la longitud especificada
ncurses_mvaddstr -- Mover la posición y agregar una cadena
ncurses_mvcur -- Mover el cursor inmediatamente
ncurses_mvdelch -- Mover la posición y eliminar un caracter, desplazar el resto de la línea hacia la izquierda
ncurses_mvgetch -- Mover la posición y obtener el caracter en la nueva posición
ncurses_mvhline -- Establecer una nueva posición y dibujar una línea horizontal usando un caracter con atributos y una longitud máxima de n caracteres
ncurses_mvinch -- Mover la posición y obtener el caracter con atributos en la nueva posición
ncurses_mvvline -- Establecer una nueva posición y dibujar una línea vertical usando un caracter con atributos y una longitud máxima de n caracteres
ncurses_mvwaddstr -- Agregar una cadena en una nueva posición al interior de una ventana
ncurses_napms -- Dormir
ncurses_new_panel -- Crear un nuevo panel y asociarlo con una ventana
ncurses_newpad -- Crea un nuevo pad (ventana)
ncurses_newwin -- Crear una nueva ventana
ncurses_nl -- Traducir línea nueva y retorno de carro / alimentación de línea
ncurses_nocbreak -- Cambiar la terminal a modo normal (cooked)
ncurses_noecho -- Desactivar la repetición de entrada del teclado
ncurses_nonl -- No traducir línea nueva y retorno de carro / alimentación de línea
ncurses_noqiflush -- No realizar volcados en caracteres de señales
ncurses_noraw -- Salir del modo puro en la terminal
ncurses_pair_content -- Obtiene el valor RGB del color
ncurses_panel_above -- Devuelve el panel encima del panel indicado
ncurses_panel_below -- Devuelve el panel por debajo del panel indicado
ncurses_panel_window -- Devuelve la ventana asociada con el panel
ncurses_pnoutrefresh -- Copia una región desde un pad a la pantalla virtual
ncurses_prefresh -- Copia una región desde un pad a la pantalla virtual
ncurses_putp -- Aplicar la información de márgenes a la cadena e imprimirla
ncurses_qiflush -- Producir volcado en caracteres de señales
ncurses_raw -- Colocar la terminal en modo puro
ncurses_refresh -- Refrescar la pantalla
ncurses_replace_panel -- Reemplaza la ventana asociada con el panel
ncurses_reset_prog_mode -- Restablece el modo de programa guardado por def_prog_mode
ncurses_reset_shell_mode -- Restablece el modo de intérprete de comandos guardado por def_shell_mode
ncurses_resetty -- Restablece el estado guardado de la terminal
ncurses_savetty -- Guarda el estado de la terminal
ncurses_scr_dump -- Vuelca el contenido de la pantalla en un archivo
ncurses_scr_init -- Inicializar la pantalla desde un volcado en un archivo
ncurses_scr_restore -- Recuperar la pantalla desde un volcado en un archivo
ncurses_scr_set -- Heredar la pantalla de un volcado en un archivo

ncurses_scrl -- Desplazar el contenido de la ventana hacia arriba o abajo sin cambiar la posición actual
ncurses_show_panel -- Coloca un panel invisible al comienzo de la pila, haciéndola visible
ncurses_slk_attr -- Devuelve el atributo de la etiqueta suave de teclado
ncurses_slk_attroff -- Deshabilitar los atributos dados para las etiquetas suaves de teclado
ncurses_slk_attron -- Habilitar los atributos dados para las etiquetas suaves de funciones de teclado
ncurses_slk_attrset -- Definir los atributos dados para las etiquetas suaves de funciones de teclado
ncurses_slk_clear -- Limpia las etiquetas suaves de la pantalla
ncurses_slk_color -- Establece el color para las etiquetas suaves de teclado
ncurses_slk_init -- Inicializa las etiquetas suaves de funciones de teclado
ncurses_slk_noutrefresh -- Copia las etiquetas suaves de teclado a la pantalla virtual
ncurses_slk_refresh -- Copia las etiquetas suaves de teclado a la pantalla
ncurses_slk_restore -- Recupera las etiquetas suaves de teclado
ncurses_slk_set -- Establece las etiquetas de teclado
ncurses_slk_touch -- Obliga a que se produzca salida cuando se ejecute ncurses_slk_noutrefresh
ncurses_standend -- Dejar de usar el atributo 'standout'
ncurses_standout -- Comenzar a usar el atributo 'standout'
ncurses_start_color -- Comenzar a usar colores
ncurses_termattrs -- Devuelve un valor OR lógico de todas las banderas de atributos soportados por la terminal
ncurses_termname -- Devuelve el nombre (corto) de la terminal
ncurses_timeout -- Establecer el tiempo de espera para secuencias especiales de teclas
ncurses_top_panel -- Mueve un panel visible al tope de la pila
ncurses_typeahead -- Especificar un descriptor de archivo diferente para el chequeo de escritura siguiente
ncurses_ungetch -- Colocar un caracter de vuelta en la secuencia de entrada
ncurses_ungetmouse -- Coloca un evento de mouse en la cola
ncurses_update_panels -- Refresca la pantalla virtual, para reflejar las relaciones entre los paneles en la pila
ncurses_use_default_colors -- Asignar los colores predeterminados de la terminal al id de color -1
ncurses_use_env -- Controlar el uso de información del entorno sobre el tamaño de la terminal
ncurses_use_extended_names -- Controlar el uso de nombres extendidos en descripciones de información de terminal
ncurses_vidattr -- Desplegar la cadena en la terminal con el atributo de modo de video
ncurses_vline -- Dibujar una línea vertical en la posición actual usando un caracter con atributos y una longitud máxima de n caracteres
ncurses_waddch -- Agrega un caracter en la posición actual de una ventana y avanza el cursor
ncurses_waddstr -- Imprime un texto en la posición actual de una ventana
ncurses_wattroff -- Deshabilita los atributos para una ventana
ncurses_wattron -- Habilita los atributos para una ventana
ncurses_wattrset -- Establecer los atributos para una ventana
ncurses_wborder -- Dibuja un borde alrededor de la ventana usando caracteres con atributos
ncurses_wclear -- Limpia la ventana
ncurses_wcolor_set -- Establece parejas de colores en la ventana
ncurses_werase -- Eliminar los contenidos de la ventana
ncurses_wgetch -- Lee un caracter desde el teclado (en la ventana)
ncurses_whline -- Dibuja una línea horizontal en una ventana en la posición actual usando un caracter con atributos y un largo máximo de n caracteres
ncurses_wmouse_trafo -- Transforma coordenadas de ventana/stdscr
ncurses_wmove -- Mueve la posición de salida de la ventana
ncurses_wnoutrefresh -- Copia la ventana a la pantalla virtual
ncurses_wrefresh -- Refrescar la ventana en la pantalla de terminal
ncurses_wstandend -- Finaliza el modo standout para una ventana
ncurses_wstandout -- Ingresar a modo standout en una ventana
ncurses_wvline -- Dibuja una línea vertical en una ventana en la posición actual usando un caracter

con atributos y una longitud máxima de n caracteres

ncurses_addch

(PHP 4 >= 4.1.0, PHP 5)

ncurses_addch -- Agregar un caracter en la posición actual y avanzar el cursor

Descripción

int **ncurses_addch** (int car)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_addchnstr

(PHP 4 >= 4.2.0, PHP 5)

ncurses_addchnstr -- Agregar cadena con atributos y longitud especificada en la posición actual

Descripción

int **ncurses_addchnstr** (string s, int n)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_addchstr

(PHP 4 >= 4.2.0, PHP 5)

ncurses_addchstr -- Agregar una cadena con atributos en la posición actual

Descripción

int **ncurses_addchstr** (string s)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_addnstr

(PHP 4 >= 4.2.0, PHP 5)

ncurses_addnstr -- Agregar una cadena con la longitud especificada en la posición actual

Descripción

int **ncurses_addnstr** (string s, int n)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_addstr

(PHP 4 >= 4.2.0, PHP 5)

ncurses_addstr -- Imprimir texto en la posición actual

Descripción

int **ncurses_addstr** (string texto)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_assume_default_colors

(PHP 4 >= 4.2.0, PHP 5)

ncurses_assume_default_colors -- Definir colores predeterminados para el color 0

Descripción

int **ncurses_assume_default_colors** (int fg, int bg)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_attroff

(PHP 4 >= 4.1.0, PHP 5)

ncurses_attroff -- Deshabilitar los atributos dados

Descripción

int **ncurses_attroff** (int atributos)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_attron

(PHP 4 >= 4.1.0, PHP 5)

ncurses_attron -- Habilitar los atributos dados

Descripción

int **ncurses_attron** (int atributos)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_attrset

(PHP 4 >= 4.1.0, PHP 5)

ncurses_attrset -- Establecer los atributos dados

Descripción

int **ncurses_attrset** (int atributos)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_baudrate

(PHP 4 >= 4.1.0, PHP 5)

ncurses_baudrate -- Devuelve la tasa de baudios de la terminal

Descripción

int **ncurses_baudrate** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_beep

(PHP 4 >= 4.1.0, PHP 5)

ncurses_beep -- Producir un beep en la terminal

Descripción

int **ncurses_beep** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

ncurses_beep() envía una alerta auditiva (campana), y si no es posible produce un relampagueo en la pantalla. Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Vea también **ncurses_flash()**

ncurses_bkgd

(PHP 4 >= 4.1.0, PHP 5)

ncurses_bkgd -- Establecer la propiedad de segundo plano para la pantalla de la terminal

Descripción

int **ncurses_bkgd** (int atributo_caracter)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_bkgdset

(PHP 4 >= 4.1.0, PHP 5)

ncurses_bkgdset -- Controla el segundo plano de la pantalla

Descripción

void **ncurses_bkgdset** (int atributo_caracter)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_border

(PHP 4 >= 4.2.0, PHP 5)

ncurses_border -- Dibujar un borde alrededor de la pantalla usando caracteres con atributos

Descripción

int **ncurses_border** (int izquierda, int derecha, int arriba, int abajo, int esquina_sup_izq, int esquina_sup_der, int esquina_inf_izq, int esquina_inf_der)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

ncurses_border() dibuja las líneas y esquinas especificadas alrededor de la ventana principal. Cada parámetro espera 0 para dibujar una línea, y 1 para saltarla. Las esquinas son la superior izquierda, la superior derecha, la inferior izquiera y la inferior derecha.

¡Utilice **ncurses_wborder()** para los bordes alrededor de sub-ventanas!

Vea también **ncurses_wborder()**.

ncurses_bottom_panel

(PHP 4 >= 4.3.0, PHP 5)

ncurses_bottom_panel -- Mueve un panel visible al fondo de la pila

Descripción

int **ncurses_bottom_panel** (resource panel)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_can_change_color

(PHP 4 >= 4.1.0, PHP 5)

ncurses_can_change_color -- Chequear si es posible cambiar los colores de la terminal

Descripción

bool **ncurses_can_change_color** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

La función **ncurses_can_change_color()** devuelve **`TRUE`** o **`FALSE`**, dependiendo de que la terminal soporte colores, y si el programador puede modificar los colores.

ncurses_cbreak

(PHP 4 >= 4.1.0, PHP 5)

ncurses_cbreak -- Cambio de búferes de entrada

Descripción

bool **ncurses_cbreak** (void)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

ncurses_cbreak() deshabilita el uso de búferes de línea y el procesamiento de caracteres (los caracteres de interrupción y control de flujo no son alterados), haciendo que los caracteres escritos por el usuario sean entregados al programa inmediatamente.

ncurses_cbreak() devuelve **`TRUE`** o NCURSES_ERR si ocurre un error.

Vea también: **ncurses_nocbreak()**

ncurses_clear

(PHP 4 >= 4.1.0, PHP 5)

ncurses_clear -- Limpiar la pantalla

Descripción

bool **ncurses_clear** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

ncurses_clear() limpia la pantalla completamente sin usar caracteres en blanco. Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Nota: **ncurses_clear()** limpia la pantalla sin usar caracteres en blanco, que tienen la misma apariencia del segundo plano actual. Para limpiar la pantalla con caracteres en blanco, use **ncurses_erase()**.

Vea también **ncurses_erase()**.

ncurses_clrtobot

(PHP 4 >= 4.1.0, PHP 5)

ncurses_clrtobot -- Limpiar la pantalla desde la posición actual al final

Descripción

bool **ncurses_clrtobot** (void)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

ncurses_clrtobot() elimina todas las líneas desde el cursor hasta el fin de la pantalla y crea caracteres en blanco. Los caracteres en blanco creados por **ncurses_clrtobot()** tienen la apariencia del segundo plano actual. Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Vea también **ncurses_clear()**, y **ncurses_clrtoeol()**

ncurses_clrtoeol

(PHP 4 >= 4.1.0, PHP 5)

ncurses_clrtoeol -- Limpiar la pantalla desde la posición actual al final de la línea

Descripción

bool **ncurses_clrtoeol** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

ncurses_clrtoeol() elimina la línea actual desde la posición actual hasta el final. Los caracteres en blanco creados por **ncurses_clrtoeol()** tienen la apariencia del segundo plano actual. Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Vea también **ncurses_clear()**, y **ncurses_clrtobot()**

ncurses_color_content

(PHP 4 >= 4.3.0, PHP 5)

ncurses_color_content -- Obtiene el valor RGB del color

Descripción

int **ncurses_color_content** (int color, int &r, int &g, int &b)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_color_set

(PHP 4 >= 4.1.0, PHP 5)

ncurses_color_set -- Establecer los colores de primer y segundo plano

Descripción

int **ncurses_color_set** (int pareja)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_curs_set

(PHP 4 >= 4.1.0, PHP 5)

ncurses_curs_set -- Establecer el estado del cursor

Descripción

int **ncurses_curs_set** (int visibilidad)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_def_prog_mode

(PHP 4 >= 4.1.0, PHP 5)

ncurses_def_prog_mode -- Guarda los modos de terminal (programa)

Descripción

bool **ncurses_def_prog_mode** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

ncurses_def_prog_mode() guarda los modos de terminal actuales para programa (en curses) para su uso por **ncurses_reset_prog_mode()**. Devuelve **FALSE** en caso de éxito, o **TRUE** de otra forma.

Vea también: **ncurses_reset_prog_mode()**

ncurses_def_shell_mode

(PHP 4 >= 4.1.0, PHP 5)

ncurses_def_shell_mode -- Guarda los modos de terminal (intérprete de comandos)

Descripción

bool **ncurses_def_shell_mode** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

ncurses_def_shell_mode() guarda los modos de terminal actuales de intérprete de comandos (no en curses) para su uso con **ncurses_reset_shell_mode()**. Devuelve **FALSE** en caso de éxito, o **TRUE** de lo contrario.

Vea también: **ncurses_reset_shell_mode()**

ncurses_define_key

(PHP 4 >= 4.2.0, PHP 5)

ncurses_define_key -- Definir un código de tecla

Descripción

int **ncurses_define_key** (string definicion, int codigo_tecla)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_del_panel

(PHP 4 >= 4.3.0, PHP 5)

ncurses_del_panel -- Remover un panel de la pila y eliminarlo (pero no la ventana asociada)

Descripción

int **ncurses_del_panel** (resource panel)

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_delay_output

(PHP 4 >= 4.1.0, PHP 5)

ncurses_delay_output -- Retrasar la salida en la terminal usando caracteres de relleno

Descripción

int **ncurses_delay_output** (int milisegundos)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_delch

(PHP 4 >= 4.1.0, PHP 5)

ncurses_delch -- Eliminar un caracter en la posición actual, mover el resto de la línea hacia la

izquierda

Descripción

bool **ncurses_delch** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

ncurses_delch() elimina el caracter bajo el cursor. Todos los caracteres a la derecha del cursor en la misma línea son desplazados una posición hacia la izquierda, y el último caracter en la línea es llenado con un caracter en blanco. La posición del cursor no cambia. Devuelve **FALSE** en caso de tener éxito, o **TRUE** de lo contrario.

Vea también: **ncurses_deleteln()**

ncurses_deleteln

(PHP 4 >= 4.1.0, PHP 5)

ncurses_deleteln -- Eliminar una línea en la posición actual, mover el resto de la pantalla hacia arriba

Descripción

bool **ncurses_deleteln** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

ncurses_deleteln() elimina la línea actual bajo la posición del cursor. Todas las líneas por debajo de la línea actual son desplazadas una línea hacia arriba. La línea más baja es limpiada. La posición del cursor no cambia. Devuelve **FALSE** en caso de éxito, **TRUE** de lo contrario.

Vea también: **ncurses_delch()**

ncurses_delwin

(PHP 4 >= 4.1.0, PHP 5)

ncurses_delwin -- Eliminar una ventana ncurses

Descripción

int **ncurses_delwin** (resource ventana)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_doupdate

(PHP 4 >= 4.1.0, PHP 5)

ncurses_doupdate -- Escribir todas las actualizaciones preparadas sobre la terminal

Descripción

bool **ncurses_doupdate** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

ncurses_doupdate()() compara la pantalla virtual con la pantalla física y actualiza la pantalla física. De esta forma es más efectivo que usando múltiples llamadas de actualización. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

ncurses_echo

(PHP 4 >= 4.1.0, PHP 5)

ncurses_echo -- Activar la repetición de entrada del teclado

Descripción

bool **ncurses_echo** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

ncurses_echo() habilita el modo de repetición. Todos los caracteres escritos por el usuario son repetidos por **ncurses_getch()**. Devuelve **FALSE** de tener éxito, o **TRUE** si ocurrió algún error.

Para deshabilitar el modo de repetición, use **ncurses_noecho()**.

ncurses_echochar

(PHP 4 >= 4.1.0, PHP 5)

ncurses_echochar -- Salida de un caracter sencillo, incluyendo actualización

Descripción

int **ncurses_echochar** (int caracter)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_end

(PHP 4 >= 4.1.0, PHP 5)

ncurses_end -- Dejar de usar ncurses, limpiar la pantalla

Descripción

int **ncurses_end** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_erase

(PHP 4 >= 4.1.0, PHP 5)

ncurses_erase -- Limpiar pantalla de terminal

Descripción

bool **ncurses_erase** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

ncurses_erase() llena la pantalla de terminal con caracteres en blanco. Los caracteres creados tienen la apariencia del segundo plano actual, establecida por **ncurses_bkgd()**. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Vea también **ncurses_bkgd()**, y **ncurses_clear()**

ncurses_erasechar

(PHP 4 >= 4.1.0, PHP 5)

ncurses_erasechar -- Devuelve el caracter de borrado actual

Descripción

string **ncurses_erasechar** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

ncurses_erasechar() devuelve el caracter de borrado actual.

Vea también: **ncurses_killchar()**

ncurses_filter

(PHP 4 >= 4.1.0, PHP 5)

ncurses_filter -- Definir LINES para iniscr() y newterm() a 1

Descripción

int **ncurses_filter** (void)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_flash

(PHP 4 >= 4.1.0, PHP 5)

ncurses_flash -- Relampaguear la pantalla de terminal (campana visual)

Descripción

bool **ncurses_flash** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

ncurses_flash() relampaguea la pantalla, y si no es posible, envía una aletra auditiva (campana). Devuelve **`FALSE`** de tener éxito, **`TRUE`** de lo contrario.

Vea también: **ncurses_beep()**

ncurses_flushinp

(PHP 4 >= 4.1.0, PHP 5)

ncurses_flushinp -- Volcar el búfer de entrada de teclado

Descripción

bool **ncurses_flushinp** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

La función **ncurses_flushinp()** envía cualquier contenido que haya sido escrito y no haya sido leido aun por su programa. Devuelve **`FALSE`** en caso de tener éxito, **`TRUE`** de lo contrario.

ncurses_getch

(PHP 4 >= 4.1.0, PHP 5)

ncurses_getch -- Leer un caracter desde el teclado

Descripción

int **ncurses_getch** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_getmaxyx

(PHP 4 >= 4.3.0, PHP 5)

ncurses_getmaxyx -- Devuelve el tamaño de una ventana

Descripción

void **ncurses_getmaxyx** (resource ventana, int &y, int &x)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

ncurses_getmaxyx() coloca los tamaños horizontal y vertical de la ventana *ventana* en las variables *&y* y *&x* dadas. Las variables deben ser pasadas como referencias, de modo que sean actualizadas cuando el usuario cambie el tamaño de la terminal.

ncurses_getmouse

(PHP 4 >= 4.2.0, PHP 5)

ncurses_getmouse -- Lee un evento del mouse

Descripción

bool **ncurses_getmouse** (array &evento_mouse)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

ncurses_getmouse() lee un evento de mouse desde la cola. La función **ncurses_getmouse()** devolverá **FALSE** si un evento del mouse es visible en la ventana dada, o **TRUE** de lo contrario. Las opciones del evento serán entregadas en el parámetro *evento_mouse*, el cual debe ser una matriz, pasada por referencia (vea el ejemplo siguiente). En caso de éxito, se entregará una matriz asociativa con las siguientes claves:

- "id" : Id para distinguir múltiples dispositivos
- "x" : posición x de pantalla relativa en celdas de caracter
- "y" : posición y de pantalla relativa en celdas de caracter
- "z" : no soportado por el momento
- "mmask" : acción del mouse

Ejemplo 1. Ejemplo de ncurses_getmouse()

```
<?php
switch (ncurses_getch()){
  case NCURSES_KEY_MOUSE:
    if (!ncurses_getmouse(&$mevent)){
      if ($mevent["mmask"] & NCURSES_MOUSE_BUTTON1_PRESSED){
        $mouse_x = $mevent["x"]; // Guardar la posicion del mouse
        $mouse_y = $mevent["y"];
      }
    }
  break;

  default:
    /* .... */
}
?>
```

Vea también **ncurses_ungetmouse()**

ncurses_getyx

(PHP 4 >= 4.3.0, PHP 5)

ncurses_getyx -- Devuelve la posición de cursor actual para una ventana

Descripción

void **ncurses_getyx** (resource ventana, int &y, int &x)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_halfdelay

(PHP 4 >= 4.1.0, PHP 5)

ncurses_halfdelay -- Colocar la terminal en modo de medio retraso

Descripción

int **ncurses_halfdelay** (int decimo)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_has_colors

(PHP 4 >= 4.1.0, PHP 5)

ncurses_has_colors -- Chequear si la terminal tiene colores

Descripción

bool **ncurses_has_colors** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

ncurses_has_colors() devuelve **`TRUE`** o **`FALSE`** dependiendo de que la terminal tenga soporte de colores.

Vea también: **ncurses_can_change_color()**

ncurses_has_ic

(PHP 4 >= 4.1.0, PHP 5)

ncurses_has_ic -- Chequear por el soporte de inserción y borrado

Descripción

bool **ncurses_has_ic** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

ncurses_has_ic() chequea el soporte de inserción y borrado de la terminal. Devuelve **TRUE** cuando la terminal tiene soporte de inserción/borrado, **FALSE** de lo contrario.

Vea también: **ncurses_has_il()**

ncurses_has_il

(PHP 4 >= 4.1.0, PHP 5)

ncurses_has_il -- Chequea el soporte de inserción y borrado

Descripción

bool **ncurses_has_il** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

ncurses_has_il() chequea el soporte de inserción y borrado de líneas de la terminal. Devuelve **TRUE** cuando la terminal tiene soporte de inserción/borrado de líneas, **FALSE** de lo contrario.

Vea también: **ncurses_has_ic()**

ncurses_has_key

(PHP 4 >= 4.1.0, PHP 5)

ncurses_has_key -- Chequear por la presencia de una tecla de función en el teclado de la terminal

Descripción

int **ncurses_has_key** (int codigo_tecla)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_hide_panel

(PHP 4 >= 4.3.0, PHP 5)

ncurses_hide_panel -- Remover un panel de la pila, haciéndolo invisible

Descripción

int **ncurses_hide_panel** (resource panel)

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_hline

(PHP 4 >= 4.2.0, PHP 5)

ncurses_hline -- Dibujar una línea horizontal en la posición actual usando un caracter con atributos, y de máximo n caracteres de largo

Descripción

int **ncurses_hline** (int caracter_atributos, int n)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_inch

(PHP 4 >= 4.1.0, PHP 5)

ncurses_inch -- Obtener el caracter y los atributos de la posición actual

Descripción

string **ncurses_inch** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

ncurses_inch() devuelve el caracter de la posición actual.

ncurses_init_color

(PHP 4 >= 4.2.0, PHP 5)

ncurses_init_color -- Establecer un nuevo valor RGB para un color

Descripción

int **ncurses_init_color** (int color, int r, int g, int b)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_init_pair

(PHP 4 >= 4.1.0, PHP 5)

ncurses_init_pair -- Reservar una pareja de color

Descripción

int **ncurses_init_pair** (int pareja, int fg, int bg)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_init

(PHP 4 >= 4.1.0, PHP 5)

ncurses_init -- Inicializar ncurses

Descripción

int **ncurses_init** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

ncurses_init() inicializa la interfaz ncurses y debe ser usada antes de cualquier otra función de ncurses.

ncurses_insch

(PHP 4 >= 4.1.0, PHP 5)

ncurses_insch -- Insertar un caracter, moviendo el resto de la línea, incluyendo el caracter en la posición actual

Descripción

int **ncurses_insch** (int caracter)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.
Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_insdelln

(PHP 4 >= 4.1.0, PHP 5)

ncurses_insdelln -- Insertar líneas antes de la línea actual, desplazando hacia abajo (los números

negativos eliminan líneas y desplazan hacia arriba)

Descripción

int **ncurses_insdelln** (int conteo)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.
Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_insertln

(PHP 4 >= 4.1.0, PHP 5)

ncurses_insertln -- Insertar una línea, mover el resto de la pantalla hacia abajo

Descripción

bool **ncurses_insertln** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

ncurses_insertln() inserta una nueva línea antes de la línea actual. La línea más baja se perderá.

ncurses_insstr

(PHP 4 >= 4.2.0, PHP 5)

ncurses_insstr -- Insertar cadena en la posición actual, moviendo el resto de la línea hacia la derecha

Descripción

int **ncurses_insstr** (string texto)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_instr

(PHP 4 >= 4.2.0, PHP 5)

ncurses_instr -- Lee una cadena desde la pantalla de terminal

Descripción

int **ncurses_instr** (string &bufer)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

ncurses_instr() devuelve el número de caracteres leidos desde la posición del caracter actual hasta el fin de línea. *bufer* contiene los caracteres. Los atributos son retirados de los caracteres.

ncurses_isendwin

(PHP 4 >= 4.1.0, PHP 5)

ncurses_isendwin -- Ncurses se encuentra en modo endwin, puede efectuarse salida de pantalla normal

Descripción

bool **ncurses_isendwin** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

ncurses_isendwin() devuelve **`TRUE`** si la función **ncurses_endwin()** ha sido llamada sin llamada subsiguiente alguna a **ncurses_wrefresh()**, o **`FALSE`** de lo contrario.

Vea también **ncurses_endwin()** y **ncurses_wrefresh()**.

ncurses_keyok

(PHP 4 >= 4.2.0, PHP 5)

ncurses_keyok -- Habilitar o deshabilitar un código de tecla

Descripción

int **ncurses_keyok** (int codigo_tecla, bool habilitar)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_keypad

(PHP 4 >= 4.2.0, PHP 5)

ncurses_keypad -- Activa o desactiva el teclado numérico

Descripción

int **ncurses_keypad** (resource ventana, bool bf)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_killchar

(PHP 4 >= 4.1.0, PHP 5)

ncurses_killchar -- Devuelve el caracter de eliminación de línea actual

Descripción

bool **ncurses_killchar** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

ncurses_killchar() devuelve el caracter de eliminación de línea actual.

Vea también: **ncurses_erasechar()**

ncurses_longname

(PHP 4 >= 4.2.0, PHP 5)

ncurses_longname -- Devuelve la descripción de la terminal

Descripción

string **ncurses_longname** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

ncurses_longname() devuelve una descripción detallada de la terminal. La descripción es truncada a 128 caracteres. En caso de fallo, **ncurses_longname()** devuelve NULL.

Vea también: **ncurses_termname()**

ncurses_meta

(PHP 4 >= 4.3.0, PHP 5)

ncurses_meta -- Habilita/deshabilita la meta-información de 8-bits de tecla

Descripción

int **ncurses_meta** (resource ventana, bool 8bit)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_mouse_trafo

(PHP 4 >= 4.2.0, PHP 5)

ncurses_mouse_trafo -- Transforma coordenadas

Descripción

bool **ncurses_mouse_trafo** (int &y, int &x, bool a_pantalla)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_mouseinterval

(PHP 4 >= 4.1.0, PHP 5)

ncurses_mouseinterval -- Establecer el tiempo de espera para clics de botón del mouse

Descripción

int **ncurses_mouseinterval** (int milisegundos)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_mousemask

(PHP 4 >= 4.2.0, PHP 5)

ncurses_mousemask -- Establece opciones del mouse

Descripción

int **ncurses_mousemask** (int nueva_mascara, int &vieja_mascara)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

La función **ncurses_mousemask()** establecerá los eventos del mouse a ser reportados. Por defecto, ningún evento del mouse será reportado. La función **ncurses_mousemask()** devolverá una máscara para indicar cúales de los eventos especificados en el parámetro *nueva_mascara* pueden ser reportados. En caso de un total fracaso, devuelve 0. En el parámetro *vieja_mascara*, el cual es pasado por referencia, **ncurses_mousemask()** devuelve el valor previo de la máscara de eventos del mouse. Los eventos del mouse son representados por NCURSES_KEY_MOUSE en la secuencia de entrada **ncurses_wgetch()**. Para leer los datos del evento y recuperar el evento de la cola, llame a **ncurses_getmouse()**.

Como efecto secundario, definir una máscara del mouse como 0 en *nueva_mascara* deshabilita el puntero del mouse. Definir un valor distinto de cero habilita el puntero del mouse.

Las opciones de máscara del mouse pueden definirse con las siguientes constantes predefinidas:

- NCURSES_BUTTON1_PRESSED
- NCURSES_BUTTON1_RELEASED
- NCURSES_BUTTON1_CLICKED
- NCURSES_BUTTON1_DOUBLE_CLICKED
- NCURSES_BUTTON1_TRIPLE_CLICKED
- NCURSES_BUTTON2_PRESSED
- NCURSES_BUTTON2_RELEASED
- NCURSES_BUTTON2_CLICKED
- NCURSES_BUTTON2_DOUBLE_CLICKED
- NCURSES_BUTTON2_TRIPLE_CLICKED
- NCURSES_BUTTON3_PRESSED
- NCURSES_BUTTON3_RELEASED
- NCURSES_BUTTON3_CLICKED
- NCURSES_BUTTON3_DOUBLE_CLICKED
- NCURSES_BUTTON3_TRIPLE_CLICKED
- NCURSES_BUTTON4_PRESSED
- NCURSES_BUTTON4_RELEASED
- NCURSES_BUTTON4_CLICKED
- NCURSES_BUTTON4_DOUBLE_CLICKED
- NCURSES_BUTTON4_TRIPLE_CLICKED
- NCURSES_BUTTON_SHIFT>
- NCURSES_BUTTON_CTRL
- NCURSES_BUTTON_ALT
- NCURSES_ALL_MOUSE_EVENTS
- NCURSES_REPORT_MOUSE_POSITION

Ejemplo 1. Ejemplo de ncurses_mousemask()

```
<?php
$nueva_mascara = NCURSES_BUTTON1_CLICKED + NCURSES_BUTTON1_RELEASED;
$mascara = ncurses_mousemask($nueva_mascara, &$vieja_mascara);
if ($mascara & $nueva_mascara){
  printf ("Todas las opciones del mouse especificadas ser&aacute;n soportadas\n");
}
?>
```

Vea también **ncurses_getmouse()**, **ncurses_ungetmouse()** y **ncurese_getch()**.

ncurses_move_panel

(PHP 4 >= 4.3.0, PHP 5)

ncurses_move_panel -- Mueve un panel de modo que su esquina superior izquierda se encuentre en [comienzo_x, comienzo_y]

Descripción

int **ncurses_move_panel** (resource panel, int comienzo_x, int comienzo_y)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_move

(PHP 4 >= 4.1.0, PHP 5)

ncurses_move -- Mover la posición de salida

Descripción

int **ncurses_move** (int y, int x)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_mvaddch

(PHP 4 >= 4.2.0, PHP 5)

ncurses_mvaddch -- Mover la posición actual y agregar un caracter

Descripción

int **ncurses_mvaddch** (int y, int x, int c)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_mvaddchnstr

(PHP 4 >= 4.2.0, PHP 5)

ncurses_mvaddchnstr -- Mover la posición y agregar una cadena con atributos con la longitud especificada

Descripción

int **ncurses_mvaddchnstr** (int y, int x, string s, int n)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_mvaddchstr

(PHP 4 >= 4.2.0, PHP 5)

ncurses_mvaddchstr -- Mover la posición y agregar una cadena con atributos

Descripción

int **ncurses_mvaddchstr** (int y, int x, string s)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_mvaddnstr

(PHP 4 >= 4.2.0, PHP 5)

ncurses_mvaddnstr -- Mover la posición y agregar una cadena con la longitud especificada

Descripción

int **ncurses_mvaddnstr** (int y, int x, string s, int n)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_mvaddstr

(PHP 4 >= 4.2.0, PHP 5)

ncurses_mvaddstr -- Mover la posición y agregar una cadena

Descripción

int **ncurses_mvaddstr** (int y, int x, string s)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_mvcur

(PHP 4 >= 4.2.0, PHP 5)

ncurses_mvcur -- Mover el cursor inmediatamente

Descripción

int **ncurses_mvcur** (int viejo_y, int viejo_x, int nuevo_y, int nuevo_x)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_mvdelch

(PHP 4 >= 4.2.0, PHP 5)

ncurses_mvdelch -- Mover la posición y eliminar un caracter, desplazar el resto de la línea hacia la izquierda

Descripción

int **ncurses_mvdelch** (int y, int x)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_mvgetch

(PHP 4 >= 4.2.0, PHP 5)

ncurses_mvgetch -- Mover la posición y obtener el caracter en la nueva posición

Descripción

int **ncurses_mvgetch** (int y, int x)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_mvhline

(PHP 4 >= 4.2.0, PHP 5)

ncurses_mvhline -- Establecer una nueva posición y dibujar una línea horizontal usando un caracter con atributos y una longitud máxima de n caracteres

Descripción

int **ncurses_mvhline** (int y, int x, int caracter_atributos, int n)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_mvinch

(PHP 4 >= 4.2.0, PHP 5)

ncurses_mvinch -- Mover la posición y obtener el caracter con atributos en la nueva posición

Descripción

int **ncurses_mvinch** (int y, int x)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_mvvline

(no version information, might be only in CVS)

ncurses_mvvline -- Establecer una nueva posición y dibujar una línea vertical usando un caracter con atributos y una longitud máxima de n caracteres

Descripción

int **ncurses_mvvline** (int y, int x, int caracter_atributos, int n)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_mvwaddstr

(PHP 4 >= 4.2.0, PHP 5)

ncurses_mvwaddstr -- Agregar una cadena en una nueva posición al interior de una ventana

Descripción

int **ncurses_mvwaddstr** (resource ventana, int y, int x, string texto)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_napms

(PHP 4 >= 4.1.0, PHP 5)

ncurses_napms -- Dormir

Descripción

int **ncurses_napms** (int milisegundos)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_new_panel

(PHP 4 >= 4.3.0, PHP 5)

ncurses_new_panel -- Crear un nuevo panel y asociarlo con una ventana

Descripción

resource **ncurses_new_panel** (resource ventana)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_newpad

(PHP 4 >= 4.3.0, PHP 5)

ncurses_newpad -- Crea un nuevo pad (ventana)

Descripción

resource **ncurses_newpad** (int filas, int columnas)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_newwin

(PHP 4 >= 4.1.0, PHP 5)

ncurses_newwin -- Crear una nueva ventana

Descripción

int **ncurses_newwin** (int filas, int columnas, int y, int x)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

ncurses_newwin() crea una nueva ventana en donde dibujar elementos. Las ventanas pueden ser posicionadas usando *x*, *y*, *filas* y *columnas*. Cuando se creen ventanas adicionales, recuerde usar **ncurses_getmaxyx()** para verificar el espacio disponible, ya que el tamaño de terminal es individual y puede variar. El valor de retorno es un ID de recurso usado para cambiar entre múltiples ventanas.

ncurses_nl

(PHP 4 >= 4.1.0, PHP 5)

ncurses_nl -- Traducir línea nueva y retorno de carro / alimentación de línea

Descripción

bool **ncurses_nl** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.
Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_nocbreak

(PHP 4 >= 4.1.0, PHP 5)

ncurses_nocbreak -- Cambiar la terminal a modo normal (cooked)

Descripción

bool **ncurses_nocbreak** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

La rutina **ncurses_nocbreak()** devuelve la terminal a modo normal (cooked). Inicialmente la terminal puede o no estar en modo cbreak, ya que el modo se hereda. De tal modo que un programa debería llamar a **ncurses_cbreak()** y **ncurses_nocbreak()** explícitamente. Devuelve **TRUE** si ocurrió un error, o **FALSE** de lo contrario.

Vea también: **ncurses_cbreak()**

ncurses_noecho

(PHP 4 >= 4.1.0, PHP 5)

ncurses_noecho -- Desactivar la repetición de entrada del teclado

Descripción

bool **ncurses_noecho** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

ncurses_noecho() previene la repetición de los caracteres escritos por el usuario. Devuelve **TRUE** si ocurre un error, o **FALSE** de lo contrario.

Vea también: **ncurses_echo()**, **ncurses_getch()**

ncurses_nonl

(PHP 4 >= 4.1.0, PHP 5)

ncurses_nonl -- No traducir línea nueva y retorno de carro / alimentación de línea

Descripción

bool **ncurses_nonl** (void)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_noqiflush

(PHP 4 >= 4.1.0, PHP 5)

ncurses_noqiflush -- No realizar volcados en caracteres de señales

Descripción

int **ncurses_noqiflush** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_noraw

(PHP 4 >= 4.1.0, PHP 5)

ncurses_noraw -- Salir del modo puro en la terminal

Descripción

bool **ncurses_noraw** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

ncurses_noraw() retira la terminal del modo puro. El modo puro (raw) es similar al modo cbrek, en tanto que los caracteres escritos son pasados inmediatamente al programa del usuario. Las diferencias consisten en que, en modo puro, los caracteres de control de interrupción, salida, suspensión y flujo son todos pasados sin ser interpretados, en lugar de generar una señal. Devuelve **`TRUE`** si ocurrió un error, o **`FALSE`** de lo contrario.

Vea también: **ncurses_raw()**, **ncurses_cbreak()**, **ncurses_nocbreak()**

ncurses_pair_content

(PHP 4 >= 4.3.0, PHP 5)

ncurses_pair_content -- Obtiene el valor RGB del color

Descripción

int **ncurses_pair_content** (int pareja, int &f, int &b)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_panel_above

(PHP 4 >= 4.3.0, PHP 5)

ncurses_panel_above -- Devuelve el panel encima del panel indicado

Descripción

int **ncurses_panel_above** (resource panel)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

Si el panel es null, devuelve el panel al fondo de la pila.

ncurses_panel_below

(PHP 4 >= 4.3.0, PHP 5)

ncurses_panel_below -- Devuelve el panel por debajo del panel indicado

Descripción

int **ncurses_panel_below** (resource panel)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

Si el panel no existe, devuelve el panel más arriba en la pila

ncurses_panel_window

(PHP 4 >= 4.3.0, PHP 5)

ncurses_panel_window -- Devuelve la ventana asociada con el panel

Descripción

int **ncurses_panel_window** (resource panel)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_pnoutrefresh

(PHP 4 >= 4.3.0, PHP 5)

ncurses_pnoutrefresh -- Copia una región desde un pad a la pantalla virtual

Descripción

int **ncurses_pnoutrefresh** (resource pad, int p_fil_min, int p_col_min, int s_fil_min, int s_col_min, int s_fil_max, int s_col_max)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_prefresh

(PHP 4 >= 4.3.0, PHP 5)

ncurses_prefresh -- Copia una región desde un pad a la pantalla virtual

Descripción

int **ncurses_prefresh** (resource pad, int p_fil_min, int p_col_min, int s_fil_min, int s_col_min, int s_fil_max, int s_col_max)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_putp

(PHP 4 >= 4.2.0, PHP 5)

ncurses_putp -- Aplicar la información de márgenes a la cadena e imprimirla

Descripción

int **ncurses_putp** (string texto)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_qiflush

(PHP 4 >= 4.1.0, PHP 5)

ncurses_qiflush -- Producir volcado en caracteres de señales

Descripción

int **ncurses_qiflush** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_raw

(PHP 4 >= 4.1.0, PHP 5)

ncurses_raw -- Colocar la terminal en modo puro

Descripción

bool **ncurses_raw** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

ncurses_raw() coloca la terminal en modo puro. El modo puro es similar al modo cbreak, en tanto que los caracteres escritos son pasados inmediatamente al programa del usuario. Las diferencias consisten en que, en modo puro, los caracteres de control de interrupción, salida, suspensión y flujo son todos pasados sin ser interpretados, en lugar de generar una señal. Devuelve **`TRUE`** si ocurre algún error, o **`FALSE`** de lo contrario.

Vea también: **ncurses_noraw()**, **ncurses_cbreak()**, **ncurses_nocbreak()**

ncurses_refresh

(PHP 4 >= 4.1.0, PHP 5)

ncurses_refresh -- Refrescar la pantalla

Descripción

int **ncurses_refresh** (int ch)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.
Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_replace_panel

(PHP 4 >= 4.3.0, PHP 5)

ncurses_replace_panel -- Reemplaza la ventana asociada con el panel

Descripción

int **ncurses_replace_panel** (resource panel, resource ventana)

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_reset_prog_mode

(PHP 4 >= 4.3.0, PHP 5)

ncurses_reset_prog_mode -- Restablece el modo de programa guardado por def_prog_mode

Descripción

int **ncurses_reset_prog_mode** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_reset_shell_mode

(PHP 4 >= 4.3.0, PHP 5)

ncurses_reset_shell_mode -- Restablece el modo de intérprete de comandos guardado por def_shell_mode

Descripción

int **ncurses_reset_shell_mode** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_resetty

(PHP 4 >= 4.1.0, PHP 5)

ncurses_resetty -- Restablece el estado guardado de la terminal

Descripción

bool **ncurses_resetty** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

La función **ncurses_resetty()** recupera el estado de la terminal, el cual fue guardado previamente al llamar **ncurses_savetty()**. Esta función siempre devuelve **FALSE**.

Vea también: **ncurses_savetty()**

ncurses_savetty

(PHP 4 >= 4.1.0, PHP 5)

ncurses_savetty -- Guarda el estado de la terminal

Descripción

bool **ncurses_savetty** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

La función **ncurses_savetty()** guarda el estado actual de la terminal. El estado de terminal guardado puede ser recuperado con la función **ncurses_resetty()**. **ncurses_savetty()** siempre devuelve **FALSE**.

Vea también: **ncurses_resetty()**

ncurses_scr_dump

(PHP 4 >= 4.2.0, PHP 5)

ncurses_scr_dump -- Vuelca el contenido de la pantalla en un archivo

Descripción

int **ncurses_scr_dump** (string nombre_archivo)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_scr_init

(PHP 4 >= 4.2.0, PHP 5)

ncurses_scr_init -- Inicializar la pantalla desde un volcado en un archivo

Descripción

int **ncurses_scr_init** (string nombre_archivo)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_scr_restore

(PHP 4 >= 4.2.0, PHP 5)

ncurses_scr_restore -- Recuperar la pantalla desde un volcado en un archivo

Descripción

int **ncurses_scr_restore** (string nombre_archivo)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_scr_set

(PHP 4 >= 4.2.0, PHP 5)

ncurses_scr_set -- Heredar la pantalla de un volcado en un archivo

Descripción

int **ncurses_scr_set** (string nombre_archivo)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_scrl

(PHP 4 >= 4.1.0, PHP 5)

ncurses_scrl -- Desplazar el contenido de la ventana hacia arriba o abajo sin cambiar la posición actual

Descripción

int **ncurses_scrl** (int conteo)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_show_panel

(PHP 4 >= 4.3.0, PHP 5)

ncurses_show_panel -- Coloca un panel invisible al comienzo de la pila, haciéndola visible

Descripción

int **ncurses_show_panel** (resource panel)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_slk_attr

(PHP 4 >= 4.1.0, PHP 5)

ncurses_slk_attr -- Devuelve el atributo de la etiqueta suave de teclado

Descripción

bool **ncurses_slk_attr** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

ncurses_slk_attr() devuelve el atributo de la etiqueta suave de teclado actual. En caso de fallo, devuelve **`TRUE`**, **`FALSE`** de lo contrario.

ncurses_slk_attroff

(PHP 4 >= 4.1.0, PHP 5)

ncurses_slk_attroff -- Deshabilitar los atributos dados para las etiquetas suaves de teclado

Descripción

int **ncurses_slk_attroff** (int arg_int)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_slk_attron

(PHP 4 >= 4.1.0, PHP 5)

ncurses_slk_attron -- Habilitar los atributos dados para las etiquetas suaves de funciones de teclado

Descripción

int **ncurses_slk_attron** (int intarg)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.
Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_slk_attrset

(PHP 4 >= 4.1.0, PHP 5)

ncurses_slk_attrset -- Definir los atributos dados para las etiquetas suaves de funciones de teclado

Descripción

int **ncurses_slk_attrset** (int intarg)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.
Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_slk_clear

(PHP 4 >= 4.1.0, PHP 5)

ncurses_slk_clear -- Limpia las etiquetas suaves de la pantalla

Descripción

bool **ncurses_slk_clear** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

La función **ncurses_slk_clear()** limpia las etiquetas suaves de teclado de la pantalla. Devuelve **`TRUE`** en caso de fallo, o **`FALSE`** de lo contrario.

ncurses_slk_color

(PHP 4 >= 4.1.0, PHP 5)

ncurses_slk_color -- Establece el color para las etiquetas suaves de teclado

Descripción

int **ncurses_slk_color** (int intarg)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_slk_init

(PHP 4 >= 4.1.0, PHP 5)

ncurses_slk_init -- Inicializa las etiquetas suaves de funciones de teclado

Descripción

bool **ncurses_slk_init** (int formato)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

La función **ncurses_slk_init()** debe ser llamada antes de **ncurses_initscr()** o de que **ncurses_newterm()** sea llamada. Si **ncurses_initscr()** usa eventualmente una línea de stdscr para emular las etiquetas suaves, entonces *formato* determina el modo en que las etiquetas son acomodadas en la pantalla. Al definir *formato* como 0, se indica un arreglo 3-2-3 de las etiquetas, 1 indica un arreglo 4-4 y 2 indica el modo estándar de PC 4-4-4, pero adicionalmente se creará una línea de índice.

ncurses_slk_noutrefresh

(PHP 4 >= 4.1.0, PHP 5)

ncurses_slk_noutrefresh -- Copia las etiquetas suaves de teclado a la pantalla virtual

Descripción

bool **ncurses_slk_noutrefresh** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.
Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_slk_refresh

(PHP 4 >= 4.1.0, PHP 5)

ncurses_slk_refresh -- Copia las etiquetas suaves de teclado a la pantalla

Descripción

bool **ncurses_slk_refresh** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

ncurses_slk_refresh() copia etiquetas suaves de teclado desde la pantalla virtual a la pantalla física. Devuelve **`TRUE`** en caso de fallo, o **`FALSE`** de lo contrario.

ncurses_slk_restore

(PHP 4 >= 4.1.0, PHP 5)

ncurses_slk_restore -- Recupera las etiquetas suaves de teclado

Descripción

bool **ncurses_slk_restore** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

La función **ncurses_slk_restore()** recupera las etiquetas suaves de teclado después de que se ha

completado un proceso **ncurses_slk_clear()**.

ncurses_slk_set

(PHP 4 >= 4.2.0, PHP 5)

ncurses_slk_set -- Establece las etiquetas de teclado

Descripción

bool **ncurses_slk_set** (int num_etiqueta, string etiqueta, int formato)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_slk_touch

(PHP 4 >= 4.1.0, PHP 5)

ncurses_slk_touch -- Obliga a que se produzca salida cuando se ejecute ncurses_slk_noutrefresh

Descripción

bool **ncurses_slk_touch** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

La función **ncurses_slk_touch()** obliga a que todas las etiquetas suaves sean desplegadas la próxima vez que se realiza un llamado a **ncurses_slk_noutrefresh()**.

ncurses_standend

(PHP 4 >= 4.1.0, PHP 5)

ncurses_standend -- Dejar de usar el atributo 'standout'

Descripción

int **ncurses_standend** (void)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_standout

(PHP 4 >= 4.1.0, PHP 5)

ncurses_standout -- Comenzar a usar el atributo 'standout'

Descripción

int **ncurses_standout** (void)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_start_color

(PHP 4 >= 4.1.0, PHP 5)

ncurses_start_color -- Comenzar a usar colores

Descripción

int **ncurses_start_color** (void)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_termattrs

(PHP 4 >= 4.1.0, PHP 5)

ncurses_termattrs -- Devuelve un valor OR lógico de todas las banderas de atributos soportados por la terminal

Descripción

bool **ncurses_termattrs** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_termname

(PHP 4 >= 4.2.0, PHP 5)

ncurses_termname -- Devuelve el nombre (corto) de la terminal

Descripción

string **ncurses_termname** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

ncurses_termname() devuelve el nombre corto de las terminales. El nombre corto es truncado a 14 caracteres. En caso de fallo, **ncurses_termname()** devuelve NULL.

Vea también: **ncurses_longname()**

ncurses_timeout

(PHP 4 >= 4.1.0, PHP 5)

ncurses_timeout -- Establecer el tiempo de espera para secuencias especiales de teclas

Descripción

void **ncurses_timeout** (int milisegundos)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_top_panel

(PHP 4 >= 4.3.0, PHP 5)

ncurses_top_panel -- Mueve un panel visible al tope de la pila

Descripción

int **ncurses_top_panel** (resource panel)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_typeahead

(PHP 4 >= 4.1.0, PHP 5)

ncurses_typeahead -- Especificar un descriptor de archivo diferente para el chequeo de escritura siguiente

Descripción

int **ncurses_typeahead** (int da)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_ungetch

(PHP 4 >= 4.1.0, PHP 5)

ncurses_ungetch -- Colocar un caracter de vuelta en la secuencia de entrada

Descripción

int **ncurses_ungetch** (int codigo_tecla)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_ungetmouse

(PHP 4 >= 4.2.0, PHP 5)

ncurses_ungetmouse -- Coloca un evento de mouse en la cola

Descripción

bool **ncurses_ungetmouse** (array evento_mouse)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

ncurses_getmouse() coloca un evento KEY_MOUSE en la cola de eventos devueltos y asocia con él los datos de estado y coordenadas relativas-a-pantalla de celda de caracter especificadas en *evento_mouse*. Las opciones del evento son especificadas en la matriz asociativa *evento_mouse*:

- "id" : Id para distinguir múltiples dispositivos
- "x" : posición x relativa a la pantalla en celdas de caracter
- "y" : posición y relativa a la pantalla en celdas de caracter
- "z" : por el momento sin soporte
- "mmask" : acción del mouse

ncurses_ungetmouse() devuelve **FALSE** en caso de éxito, o **TRUE** de lo contrario.

Vea también: **ncurses_getmouse()**

ncurses_update_panels

(PHP 4 >= 4.3.0, PHP 5)

ncurses_update_panels -- Refresca la pantalla virtual, para reflejar las relaciones entre los paneles en la pila

Descripción

void **ncurses_update_panels** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_use_default_colors

(PHP 4 >= 4.1.0, PHP 5)

ncurses_use_default_colors -- Asignar los colores predeterminados de la terminal al id de color -1

Descripción

bool **ncurses_use_default_colors** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_use_env

(PHP 4 >= 4.1.0, PHP 5)

ncurses_use_env -- Controlar el uso de información del entorno sobre el tamaño de la terminal

Descripción

void **ncurses_use_env** (bool bandera)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.
Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_use_extended_names

(PHP 4 >= 4.1.0, PHP 5)

ncurses_use_extended_names -- Controlar el uso de nombres extendidos en descripciones de información de terminal

Descripción

int **ncurses_use_extended_names** (bool bandera)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.
Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_vidattr

(PHP 4 >= 4.1.0, PHP 5)

ncurses_vidattr -- Desplegar la cadena en la terminal con el atributo de modo de video

Descripción

int **ncurses_vidattr** (int intarg)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.
Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_vline

(PHP 4 >= 4.2.0, PHP 5)

ncurses_vline -- Dibujar una línea vertical en la posición actual usando un caracter con atributos y una longitud máxima de n caracteres

Descripción

int **ncurses_vline** (int caracter_atributos, int n)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_waddch

(PHP 4 >= 4.3.0, PHP 5)

ncurses_waddch -- Agrega un caracter en la posición actual de una ventana y avanza el cursor

Descripción

int **ncurses_waddch** (resource ventana, int ch)

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_waddstr

(PHP 4 >= 4.2.0, PHP 5)

ncurses_waddstr -- Imprime un texto en la posición actual de una ventana

Descripción

int **ncurses_waddstr** (resource ventana, string cadena [, int n])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_wattroff

(PHP 4 >= 4.3.0, PHP 5)

ncurses_wattroff -- Deshabilita los atributos para una ventana

Descripción

int **ncurses_wattroff** (resource ventana, int atributos)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_wattron

(PHP 4 >= 4.3.0, PHP 5)

ncurses_wattron -- Habilita los atributos para una ventana

Descripción

int **ncurses_wattron** (resource ventana, int atributos)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_wattrset

(PHP 4 >= 4.3.0, PHP 5)

ncurses_wattrset -- Establecer los atributos para una ventana

Descripción

int **ncurses_wattrset** (resource ventana, int atributos)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_wborder

(PHP 4 >= 4.3.0, PHP 5)

ncurses_wborder -- Dibuja un borde alrededor de la ventana usando caracteres con atributos

Descripción

int **ncurses_wborder** (resource ventana, int izquierda, int derecha, int arriba, int abajo, int esq_sup_izq, int esq_sup_der, int esq_inf_izq, int esq_inf_der)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

ncurses_wborder() dibuja las líneas y esquinas indicadas alrededor de la ventana *ventana* dada. Cada parámetro espera 0 para dibujar una línea, y 1 para saltarla. Las esquinas son la superior izquierda, la superior derecha, la inferior izquierda y la inferior derecha.

Utilice **ncurses_border()** para bordes alrededor de la ventana principal.

Vea también **ncurses_border()**.

ncurses_wclear

(PHP 4 >= 4.2.0, PHP 5)

ncurses_wclear -- Limpia la ventana

Descripción

int **ncurses_wclear** (resource ventana)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_wcolor_set

(PHP 4 >= 4.2.0, PHP 5)

ncurses_wcolor_set -- Establece parejas de colores en la ventana

Descripción

int **ncurses_wcolor_set** (resource ventana, int pareja_color)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_werase

(PHP 4 >= 4.3.0, PHP 5)

ncurses_werase -- Eliminar los contenidos de la ventana

Descripción

int **ncurses_werase** (resource ventana)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_wgetch

(PHP 4 >= 4.2.0, PHP 5)

ncurses_wgetch -- Lee un caracter desde el teclado (en la ventana)

Descripción

int **ncurses_wgetch** (resource ventana)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_whline

(PHP 4 >= 4.3.0, PHP 5)

ncurses_whline -- Dibuja una línea horizontal en una ventana en la posición actual usando un caracter con atributos y un largo máximo de n caracteres

Descripción

int **ncurses_whline** (resource ventana, int caracter_atributos, int n)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_wmouse_trafo

(PHP 4 >= 4.2.0, PHP 5)

ncurses_wmouse_trafo -- Transforma coordenadas de ventana/stdscr

Descripción

bool **ncurses_wmouse_trafo** (resource ventana, int &y, int &x, bool a_pantalla)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_wmove

(PHP 4 >= 4.2.0, PHP 5)

ncurses_wmove -- Mueve la posición de salida de la ventana

Descripción

int **ncurses_wmove** (resource ventana, int y, int x)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_wnoutrefresh

(PHP 4 >= 4.2.0, PHP 5)

ncurses_wnoutrefresh -- Copia la ventana a la pantalla virtual

Descripción

int **ncurses_wnoutrefresh** (resource ventana)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_wrefresh

(PHP 4 >= 4.2.0, PHP 5)

ncurses_wrefresh -- Refrescar la ventana en la pantalla de terminal

Descripción

int **ncurses_wrefresh** (resource ventana)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_wstandend

(PHP 4 >= 4.3.0, PHP 5)

ncurses_wstandend -- Finaliza el modo standout para una ventana

Descripción

int **ncurses_wstandend** (resource ventana)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_wstandout

(PHP 4 >= 4.3.0, PHP 5)

ncurses_wstandout -- Ingresar a modo standout en una ventana

Descripción

int **ncurses_wstandout** (resource ventana)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ncurses_wvline

(PHP 4 >= 4.3.0, PHP 5)

ncurses_wvline -- Dibuja una línea vertical en una ventana en la posición actual usando un caracter con atributos y una longitud máxima de n caracteres

Descripción

int **ncurses_wvline** (resource ventana, int caracter_atributos, int n)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

LXXXII. Funciones de Red

Introducción

Requirimientos

No se necesitan bibliotecas externas para construir esta extensión

---

Instalación

No se necesita ninguna instalación para usar estas funciones, son parte del núcleo de PHP.

Configuración en tiempo de ejecución

El comportamiento de estas funciones está afectado por los valores definidos en `php.ini`.

Tabla 1. Opciones de Configuración de Red

Nombre	Predeterminado	Modificable
define_syslog_variables	"0"	PHP_INI_ALL

Para más detalles sobre las constantes PHP_INI_* y su definición, vea **ini_set()**.

A continuación se presenta una corta explicación de las directivas de configuración.

define_syslog_variables **boolean**

> Indica si deben definirse las diferentes variables syslog (p.ej. $LOG_PID, $LOG_CRON, etc.). Deshabilitar este parámetro es una buena idea desde el punto de vista del rendimiento. En tiempo de ejecución, puede definir estas variables con un llamado a **define_syslog_variables()**.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Las constantes listadas aqui están siempre disponibles a través del "núcleo PHP".

Tabla 2. Opciones de openlog()

Constante	Descripción
LOG_CONS	si hay un error durante el envío de datos al registro del sistema, escribir directamente a la consola de sistema
LOG_NDELAY	abrir la conexión con el registro inmediatamente
LOG_ODELAY	(predeterminado) retrasar la apertura de conexión hasta que el primer mensaje sea registrado
LOG_NOWAIT	
LOG_PERROR	imprimir mensajes de registro también en stderr
LOG_PID	incluir PID con cada mensaje

Tabla 3. Facilidades de openlog()

Constante	Descripción
LOG_AUTH	mensajes de seguridad/autorización (usar LOG_AUTHPRIV en su lugar en sistemas en donde tal constante está definida)
LOG_AUTHPRIV	mensajes de seguridad/autorización (privados)
LOG_CRON	daemonio de reloj (cron y at)
LOG_DAEMON	otros daemonios de sistema
LOG_KERN	mensajes de kernel
LOG_LOCAL0 ... LOG_LOCAL7	reservadas para uso local, no están disponibles en Windows
LOG_LPR	subsistema de impresión de línea
LOG_MAIL	subsistema de correo
LOG_NEWS	subsistema de noticias USENET
LOG_SYSLOG	mensajes generados internamente por syslogd
LOG_USER	mensajes genéricos de nivel de usuario
LOG_UUCP	subsistema UUCP

Tabla 4. Prioridades de syslog() (en orden descendiente)

Constante	Descripción
LOG_EMERG	el sistema es inutilizable

Constante	Descripción
LOG_ALERT	debe tomarse una acción inmediatamente
LOG_CRIT	condiciones críticas
LOG_ERR	condiciones de error
LOG_WARNING	condiciones de advertencia
LOG_NOTICE	condición normal, pero significativa
LOG_INFO	mensaje informativo
LOG_DEBUG	mensaje de nivel de depuración

Tabla 5. Opciones de dns_get_record()

Constante	Descripción
DNS_A	Recurso de Dirección IPv4
DNS_MX	Recurso de Intercambio de Correo
DNS_CNAME	Recurso Alias (Nombre Canónico)
DNS_NS	Recurso de Autoridad de Servidor de Nombres
DNS_PTR	Recurso de Apuntador
DNS_HINFO	Recurso de Información de Host (Vea los Nombres de Sistemas Operativos de IANA para consultar el significado de estos valores)
DNS_SOA	Comienzo de Recurso de Autoridad
DNS_TXT	Recurso de Texto
DNS_ANY	Cualquier Registro de Recurso. En la mayoría de sistemas, éste valor devuelve todos los registros de recurso, sin embargo, no debería confiarse en él para usos críticos. Pruebe con DNS_ALL en su lugar.
DNS_AAAA	Recurso de Dirección IPv6
DNS_ALL	Consultar el nombre de servidor iterativamente para cada tipo de registro disponible.

Tabla de contenidos

getprotobyname -- Obtener el número de protocolo asociado con el nombre de protocolo
getprotobynumber -- Obtener el nombre de protocolo asociado con un número de protocolo
getservbyname -- obtiene el nÃ‚Â£mero del puerto asociado al servicio Internet especificado
getservbyport -- obtiene el servicio Internet que correspondiente al puerto del protocolo especificado
inet_ntop -- Converts a packed internet address to a human readable representation
inet_pton -- Converts a human readable IP address to its packed in_addr representation
ip2long -- Convierte una cadena que contiene una dirección con puntos del Protocolo de Internet (IPv4) en una dirección apropiada
long2ip -- Convierte una dirección de red Internet (IPv4) a una cadena en formato estándar de Internet con puntos
openlog -- abre una conexión con el logger del sistema
pfsockopen -- Abre conexiones persistentes de dominio Internet o Unix.
socket_get_status -- Alias de **stream_get_meta_data()**
socket_set_blocking -- Alias de **stream_set_blocking()**
socket_set_timeout -- Alias de **stream_set_timeout()**
syslog -- genera un mensaje de sistema

checkdnsrr

(PHP 3, PHP 4 , PHP 5)

checkdnsrr -- Comprueba registros DNS correspondientes a nombres de máquinas en Internet o direcciones IP.

Descripción

int **checkdnsrr** (string host [, string type])

Busca en DNS entradas del tipo *type* correspondientes a *host*. Devuelve verdadero si encuentra algún registro; devuelve falso si no encuentra ninguno o sucedió algún error.

type puede ser: A, MX, NS, SOA, PTR, CNAME, o ANY. Por defecto es MX.

host puede ser o la dirección IP de la forma xxxx.xxxx.xxxx.xxxx o el nombre de la máquina.

Ver también **getmxrr()**, **gethostbyaddr()**, **gethostbyname()**, **gethostbynamel()**, y named(8) en las páginas del manual.

closelog

(PHP 3, PHP 4 , PHP 5)

closelog -- cierra la conexión con el logger del sistema

Descripción

int **closelog** (void)

closelog() cierra el descriptor que se está usando para escribir en el logger del sistema. El uso de **closelog()** es opcional.

debugger_off

(PHP 3)

debugger_off -- deshabilita el depurador interno de PHP

Descripción

int **debugger_off** (void)

Deshabilita el depurador interno de PHP. El depurador está aún en desarrollo.

debugger_on

(PHP 3)

debugger_on -- habilita el depurador interno de PHP

Descripción

int **debugger_on** (string address)

Habilita el depurador interno de PHP, conectándolo a *address*. El depurador esta aún en desarrollo.

define_syslog_variables

(PHP 3, PHP 4 , PHP 5)

define_syslog_variables -- Inicializa todas las constantes relacionadas con syslog

Descripción

void **define_syslog_variables** (void)

Inicializa todas las constantes usadas en las funciones syslog.

Vea también **openlog()**, **syslog()** y **closelog()**.

dns_check_record

(PHP 5)

dns_check_record -- Sinónimo para **checkdnsrr()**

Descripción

int **dns_check_record** (string host [, string tipo])

Chequea los registros DNS correspondientes a un nombre de servidor huésped de Internet o dirección IP dada.

dns_get_mx

(PHP 5)

dns_get_mx -- Sinónimo para **getmxrr()**

Descripción

int **dns_get_mx** (string nombre_host, array &hosts_mx [, array &peso])

Obtener registros MX correspondientes a un nombre de servidor huésped de Internet dado.

dns_get_record

(PHP 5)

dns_get_record -- Recuperar Registros de Recursos DNS asociados con un nombre de servidor huésped

Descripción

array **dns_get_record** (string nombre_host [, int tipo [, array &authns, array &addtl]])

> **Nota:** Esta función no está implementada en plataformas Windows. Pruebe con la clase PEAR Net_DNS.

Esta función devuelve una matriz de matrices asociativas. Cada matriz asociativa contiene *por lo menos* las siguientes claves:

Tabla 1. Atributos DNS básicos

Atributo	Significado
host	El registro en el espacio de nombres DNS al que hacen referencia el resto de datos asociados.
class	**dns_get_record()** devuelve únicamente registros de clase Internet y como tal, éste parámetro siempre devolverá *IN*.
type	Cadena que contiene el tipo de registro. Otros atributos adicionales serán contenidos también en la matriz resultante dependiendo del valor de type. Vea la tabla más adelante.
ttl	Valor Time To Live restante para este registro. Este *no* será igual al ttl original del registro, sino que será igual al ttl original menos cualquiera que haya sido la longitud de tiempo transcurrido desde que el servidor de nombres oficial fuera consultado.

nombre_host debe ser un nombre de servidor huésped DNS válido, como "www.example.com". Pueden generarse consultas revertidas usando la notación in-addr.arpa, pero **gethostbyaddr()** es más apropiada para la mayoría de consultas revertidas.

Por defecto, **dns_get_record()** buscará por cualquier registro de recurso asociado con *nombre_host*. Para limitar la consulta, especifique el parámetro opcional *tipo*. *tipo* puede ser cualquiera de los siguientes: **DNS_A**, **DNS_CNAME**, **DNS_HINFO**, **DNS_MX**, **DNS_NS**, **DNS_PTR**, **DNS_SOA**, **DNS_TXT**, **DNS_AAAA**, **DNS_SRV**, **DNS_NAPTR**, **DNS_A6**, **DNS_ALL** o **DNS_ANY**. El valor predeterminado es *DNS_ANY*.

> **Nota:** Debido a excentricidades en el rendimiento de libresolv entre plataformas, **DNS_ANY** no siempre devolverá todos los registros, la alternativa más lenta **DNS_ALL** recolectará todos los registros de forma más confiable.

Los opcionales tercer y cuarto argumento de esta función, *authns* y *addtl* son pasados por referencia y, si son usados, serán llenados con Registros de Recurso para los *Servidores de Nombres Oficiales*, y cualquier *Registro Adicional* respectivamente. Vea el ejemplo más adelante.

Tabla 2. Otras claves en las matrices asociativas, dependientes de 'type'

Tipo	Columnas Extra
A	*ip*: Una dirección IPv4 en notación decimal con puntos.
MX	*pri*: Prioridad del gestor de correo. Números bás bajos indican mayor prioridad. *target*: FQDN del gestor de correo. Vea también **dns_get_mx()**.
CNAME	*target*: FQDN de la ubicación en el espacio de nombres DNS de la cual es alias el registro.
NS	*target*: FQDN del servidor de nombres que es la autoridad para este nombre de host.
PTR	*target*: Ubicación dentro del espacio de nombres DNS a la que apunta este registro.
TXT	*txt*: Cadena de datos arbitraria asociada con este registro.
HINFO	*cpu*: Número IANA que indica el CPU de la máquina referenciada por este registro. *os*: Número IANA que indica el Sistema Operativo en la máquina referenciada por este registro. Vea los Nombres de Sistemas Operativos de IANA para consultar el significado de estos valores.
SOA	*mname*: FQDN de la máquina desde donde se originaron los registros de recurso. *rname*: Dirección de correo electrónico del contacto administrativo para este dominio. *serial*: # serial de esta revisión del dominio solicitado. *refresh*: Intervalo de actualización (segundos) que deben usar los servidores de nombre secundarios cuando actualicen las copias remotas de este dominio. *retry*: Periodo de tiempo (segundos) para esperar después de una actualización fallida antes de hacer un segundo intento. *expire*: Periodo máximo de tiempo (segundos) que un servidor DNS secundario debe conservar copias remotas de los datos de zona sin una actualización exitosa antes de descartarlos. *minimum-ttl*: Periodo mínimo de tiempo (segundos) en el que un cliente puede continuar usando una resolución DNS antes de que deba solicitar una nueva resolución del servidor. Puede ser sobrescrito por registros de recurso individuales.
AAAA	*ipv6*: Dirección IPv6
A6(PHP >= 5.1.0)	*masklen*: Longitud (en bits) para heredar del destino especificado por *chain*. *ipv6*: Dirección para mezclar este registro específico con *chain*. *chain*: Registro padre a mezclar con los datos de *ipv6*.
SRV	*pri*: (Prioridad) las prioridades más bajas deben ser usadas primero. *weight*: Rangos para pesar cuál de los *destinos* comunmente con prioridades debe ser elegido al azar. *target* y *port*: nombre de servidor huésped y puerto en dondo puede encontrarse el servicio solicitado. Para más información, vea: RFC 2782

Tipo	Columnas Extra
NAPTR	*order* y *pref*: ñalente a *pri* y *weight* descritos anteriormente. *flags*, *services*, *regex*, y *replacement*: Parámetros, como se definen por RFC 2915.

Nota: Debido a estándares DNS, las direcciones de correo electrónico son dadas en formato usuario.host (por ejemplo: hostmaster.example.com en lugar de hostmaster@example.com), asegúrese de chequear este valor y modificarlo si es necesario antes de usarlo con funciones como **mail()**.

Ejemplo 1. Uso de dns_get_record()

```
<?php
$resultado = dns_get_record("php.net");
print_r($resultado);
?>
```

Produce una salida similar a la siguiente:

```
Array
(
    [0] => Array
        (
            [host] => php.net
            [type] => MX
            [pri] => 5
            [target] => pair2.php.net
            [class] => IN
            [ttl] => 6765
        )

    [1] => Array
        (
            [host] => php.net
            [type] => A
            [ip] => 64.246.30.37
            [class] => IN
            [ttl] => 8125
        )

)
```

Ya que es muy común desear la dirección IP de un servidor de correo una vez el registro MX ha sido resuelto, **dns_get_record()** también devuelve una matriz en *addtl* la cual contiene registros asociados. *authns* es devuelto también, conteniendo una lista de servidores de nombre oficiales.

Ejemplo 2. Uso de dns_get_record() y DNS_ANY

```
<?php

/* Solicitar "cualquier" registro para php.net, y crear las matrices
   $authns y $addtl conteniendo una lista de servidores de nombre y
   cualquier registro adicional que vaya con ellos */

$resultado = dns_get_record("php.net", DNS_ANY, $authns, $addtl);
echo "Resultado = ";
print_r($resultado);
echo "Auth NS = ";
print_r($authns);
echo "Adicional = ";
print_r($addtl);
?>
```

Produce una salida similar a la siguiente:

```
Result = Array
(
    [0] => Array
        (
            [host] => php.net
            [type] => MX
            [pri] => 5
            [target] => pair2.php.net
            [class] => IN
            [ttl] => 6765
        )

    [1] => Array
        (
            [host] => php.net
            [type] => A
            [ip] => 64.246.30.37
            [class] => IN
            [ttl] => 8125
        )

)
Auth NS = Array
(
    [0] => Array
        (
            [host] => php.net
            [type] => NS
            [target] => remote1.easydns.com
            [class] => IN
            [ttl] => 10722
        )

    [1] => Array
        (
            [host] => php.net
            [type] => NS
            [target] => remote2.easydns.com
            [class] => IN
            [ttl] => 10722
        )

    [2] => Array
        (
            [host] => php.net
            [type] => NS
            [target] => ns1.easydns.com
            [class] => IN
            [ttl] => 10722
        )

    [3] => Array
        (
            [host] => php.net
            [type] => NS
            [target] => ns2.easydns.com
            [class] => IN
            [ttl] => 10722
        )

)
Additional = Array
(
    [0] => Array
        (
            [host] => pair2.php.net
            [type] => A
            [ip] => 216.92.131.5
            [class] => IN
            [ttl] => 6766
        )
```

Vea también **dns_get_mx()**, y **dns_check_record()**

fsockopen

(PHP 3, PHP 4 , PHP 5)

fsockopen -- Abrir una conexión de sockets de dominio de Internet o Unix

Descripción

resource **fsockopen** (string destino, int puerto [, int &errno [, string &errstr [, float tiempo_espera]]])

Inicia una conexión a través de sockets con el recurso especificado por *destino*. PHP soporta el uso de destinos en los dominios de Internet y Unix, tal y como se describe en Apéndice N. Una lista de transportes soportados puede recuperarse usando **stream_get_transports()**.

> **Nota:** Si necesita establecer un tiempo de espera para la lectura/escritura de datos a través del socket, use **stream_set_timeout()**, dado que el parámetro *tiempo_espera* de **fsockopen()** sólo se aplica cuando se conecta con el socket.

A partir de PHP 4.3.0, si ha compilado el soporte para OpenSSL, usted puede usar un prefijo sobre el *nombre_host*, ya sea '*ssl://*' o '*tls://*' para usar una conexión de cliente SSL o TLS sobre TCP/IP para conectarse con el host remoto.

fsockopen() devuelve un apuntador de archivo el cual puede ser usado junto con otras funciones de archivos (como **fgets()**, **fgetss()**, **fwrite()**, **fclose()**, y **feof()**).

Si la llamada falla, devolverá **`FALSE`**, y si los argumentos opcionales *errno* y *errstr* están presentes, serán modificados para indicar el error de nivel de sistema real que ocurrió en la llamada de sistema *connect()*. Si el valor devuelto en *errno* es *0* y la función devolvió **`FALSE`**, es un indicio de que el error ocurrió antes de la llamada a *connect()*. Lo más probable es que esto se deba a un problema con la inicialización del socket. Note que los argumentos *errno* y *errstr* siempre serán pasados por referencia.

Dependiendo del entorno, el dominio Unix o el tiempo de espera de conexión opcional pueden no estar disponibles.

El socket será abierto por defecto en modo de bloqueo. Puede cambiar a modo de no-bloqueo usando **stream_set_blocking()**.

Ejemplo 1. Ejemplo de fsockopen()

```php
<?php
$da = fsockopen("www.example.com", 80, $errno, $errstr, 30);
if (!$da) {
    echo "$errstr ($errno)<br />\n";
} else {
    $salida = "GET / HTTP/1.1\r\n";
    $salida .= "Host: www.example.com\r\n";
    $salida .= "Connection: Close\r\n\r\n";

    fwrite($da, $salida);
    while (!feof($da)) {
        echo fgets($da, 128);
    }
    fclose($da);
}
?>
```

El ejemplo a continuación presenta la forma de recuperar la fecha y hora desde el servicio UDP "daytime" (puerto 13) en su propia máquina.

Ejemplo 2. Uso de una conexión UDP

```php
<?php
$da = fsockopen("udp://127.0.0.1", 13, $errno, $errstr);
if (!$da) {
    echo "ERROR: $errno - $errstr<br />\n";
} else {
    fwrite($da, "\n");
    echo fread($da, 26);
    fclose($da);
}
?>
```

Aviso
En ocasiones, los sockets UDP parecerán haber sido abiertos sin errores, incluso si el host remoto no puede ser contactado. El error sólo se percibirá cuando lea o escriba datos hacia/desde el socket. La razón es que UDP es un protocolo "sin conexión", lo que quiere decir que el sistema operativo no intenta establecer un enlace con el socket hasta que necesite realmente enviar o recibir datos.

Nota: Cuando se especifique una dirección numérica IPv6 (p.ej. fe80::1) se debe incluir la IP entre corchetes. Por ejemplo *tcp://[fe80::1]:80*.

Nota: El parámetro de tiempo de espera fue introducido en PHP 3.0.9 y el soporte para UDP fue añadido en PHP 4.

Vea también **pfsockopen()**, **stream_set_blocking()**, **stream_set_timeout()**, **fgets()**, **fgetss()**, **fwrite()**, **fclose()**, **feof()**, y la extensión Curl.

gethostbyaddr

(PHP 3, PHP 4 , PHP 5)

gethostbyaddr -- Obtiene el nombre de una máquina en Internet mediante su dirección IP.

Descripción

string **gethostbyaddr** (string ip_address)

Devuelve el nombre del ordenador conectado a Internet espeficado por el parámetro *ip_address*. Si ocurre un error, devuelve *ip_address*.

Ver también **gethostbyname()**.

gethostbyname

(PHP 3, PHP 4 , PHP 5)

gethostbyname -- Obtiene la dirección IP correspondiente al nombre de una máquina conectada a Internet.

Descripción

string **gethostbyname** (string hostname)

Devuelve la dirección IP de una máquina conectada a Internet especificada por *hostname*.

Ver también **gethostbyaddr()**.

gethostbynamel

(PHP 3, PHP 4 , PHP 5)

gethostbynamel -- Obtiene una lista de direcciones IP correspondientea a los nombres de máquinas conectadas a Internet.

Descripción

array **gethostbynamel** (string hostname)

Devuleve una lista de direcciones IP pertenecientes a ordenadores especificados por *hostname*.

Ver también **gethostbyname()**, **gethostbyaddr()**, **checkdnsrr()**, **getmxrr()**, y named(8) en las páginas del manual.

getmxrr

(PHP 3, PHP 4 , PHP 5)

getmxrr -- Obtiene registros MX correspondientes a una máquina conectada a Internet.

Descripción

int **getmxrr** (string hostname, array mxhosts [, array weight])

Busca DNS de registros MX correspondientes a *hostname*. Devuelve verdadero si encuentra algÃ ,Â£n registro; devuelve falso si no encuentra ninguno o se produce un error.

La lista de registros MX encontrados se colocan en el array *mxhosts*. Si se proporciona el array *weight*, se rellenará con la información obtenida.

Ver también **checkdnsrr()**, **gethostbyname()**, **gethostbynamel()**, **gethostbyaddr()**, y named(8) de las páginas del manual.

getprotobyname

(PHP 4 , PHP 5)

getprotobyname -- Obtener el número de protocolo asociado con el nombre de protocolo

Descripción

int **getprotobyname** (string nombre)

getprotobyname() devuelve el número de protocolo asociado con el *nombre* de protocolo, indicado por `/etc/protocols`.

Ejemplo 1. Ejemplo de getprotobyname()

```
<?php
$protocolo = 'tcp';
$obtener_prot = getprotobyname($protocolo);
if ($obtener_prot == -1) {
    // si no se encuentra nada, devuelve -1
    echo 'Protocolo Inv&aacute;lido';
} else {
    echo 'Protocolo #' . $obtener_prot;
}
?>
```

Vea también: **getprotobynumber()**.

getprotobynumber

(PHP 4 , PHP 5)

getprotobynumber -- Obtener el nombre de protocolo asociado con un número de protocolo

Descripción

string **getprotobynumber** (int numero)

getprotobynumber() devuelve el nombre de protocolo asociado con el *numero* de procotolo, como lo indica `/etc/protocols`.

Vea también: **getprotobyname()**.

getservbyname

(PHP 4 , PHP 5)

getservbyname -- obtiene el nÃ‚Â£mero del puerto asociado al servicio Internet especificado

Descripción

int **getservbyname** (string service, string protocol)

getservbyname() devuelve el puerto que corresponde al *service* especificado por el *protocol* en /etc/services. *protocol* puede ser *tcp* o *udp*. Ver también **getservbyport()**.

getservbyport

(PHP 4 , PHP 5)

getservbyport -- obtiene el servicio Internet que correspondiente al puerto del protocolo especificado

Descripción

string **getservbyport** (int port, string protocol)

getservbyport() delvuelve el servicio Internet asociado al *port* para el *protocol* especificado en /etc/services. *protocol* puede ser *tcp* o *udp*. Ver también **getservbyname()**.

inet_ntop

(no version information, might be only in CVS)

inet_ntop -- Converts a packed internet address to a human readable representation

Description

string **inet_ntop** (string in_addr)

This function converts a 32bit IPv4, or 128bit IPv6 address (if PHP was built with IPv6 support enabled) into an address family appropriate string representation. Returns **FALSE** on failure.

Ejemplo 1. inet_ntop() Example

```
<?php
$packed = chr(127) . chr(0) . chr(0) . chr(1);
$expanded = inet_pton($packed);

/* Outputs: 127.0.0.1 */
echo $expanded;

$packed = str_repeat(chr(0), 15) . chr(1);
$expanded = inet_pton($packed);

/* Outputs: ::1 */
echo $expanded;
?>
```

See also **long2ip()**, **inet_pton()**, and **ip2long()**.

inet_pton

(no version information, might be only in CVS)

inet_pton -- Converts a human readable IP address to its packed in_addr representation

Description

string **inet_pton** (string address)

This function converts a human readable IPv4 or IPv6 address (if PHP was built with IPv6 support enabled) into an address family appropriate 32bit or 128bit binary structure.

Ejemplo 1. inet_pton() Example

```
<?php
$in_addr = inet_pton('127.0.0.1');

$in6_addr = inet_pton('::1');

?>
```

See also **ip2long()**, **inet_ntop()**, and **long2ip()**.

ip2long

(PHP 4 , PHP 5)

ip2long -- Convierte una cadena que contiene una dirección con puntos del Protocolo de Internet (IPv4) en una dirección apropiada

Descripción

int **ip2long** (string direccion_ip)

La función **ip2long()** genera una dirección de red Internet IPv4 desde su representación en formato estándar de Internet (cadena con puntos). Si *direccion_ip* es inválida, entonces se devuelve *-1*. Note que *-1* no evalúa a **FALSE** en PHP.

> **Nota:** A partir de PHP 5.0.0, **ip2long()** devuelve **FALSE** cuando *ip_address* es inválido.

Ejemplo 1. Ejemplo de ip2long()

```
<?php
$ip = gethostbyname('www.example.com');
$salida = "Las siguientes URLs son ñalentes:<br />\n";
$salida .= 'http://www.example.com/, http://' . $ip . '/, y http://' . sprintf("%u", ip
echo $salida;
?>
```

> **Nota:** Ya que el tipo entero de PHP tiene signo, y muchas direcciones IP resultarán en enteros negativos, necesita usar el especificador de formato "%u" de **sprintf()** o **printf()** para obtener la representación de cadena de la dirección IP sin signo.

Este segundo ejemplo muestra cómo imprimir una dirección convertida con la función **printf()** tanto en PHP 4 como en PHP 5:

Ejemplo 2. Desplegar una dirección IP (PHP 4)

```
<?php
$ip   = gethostbyname('www.example.com');
$long = ip2long($ip);

if ($long == -1 || $long === FALSE) {
    echo 'IP inv&aacute;lida, por favor intente de nuevo';
} else {
    echo $ip   . "\n";           // 192.0.34.166
    echo $long . "\n";           // -1073732954
    printf("%u\n", ip2long($ip)); // 3221234342
}
?>
```

ip2long() no debería usarse como la única forma de validación de IP. Combínela con **long2ip()**:

Ejemplo 3. Validación de IP

```
<?php
// asegurarse de que las IP son validas. tambien convierte una IP
// no-completa en un cuarteto debidamente separado con puntos, como
// se explica mas adelante.
$ip = long2ip(ip2long("127.0.0.1")); // "127.0.0.1"
$ip = long2ip(ip2long("10.0.0")); // "10.0.0.0"
$ip = long2ip(ip2long("10.0.256")); // "10.0.1.0"
?>
```

ip2long() trabajará también con direcciones IP no-completas. Lea http://publibn.boulder.ibm.com/doc_link/en_US/a_doc_lib/libs/commtrf2/inet_addr.htm para más información.

> **Nota: ip2long()** devolverá *-1* (PHP 4) o **`FALSE`** (PHP 5) para la IP *255.255.255.255*.

Vea también **long2ip()** y **sprintf()**.

long2ip

(PHP 4 , PHP 5)

long2ip -- Convierte una dirección de red Internet (IPv4) a una cadena en formato estándar de Internet con puntos

Descripción

string **long2ip** (int direccion_apropiada)

La función **long2ip()** genera una dirección de Internet en formato con puntos (esto quiere decir: aaa.bbb.ccc.ddd) a partir de su representación apropiada de dirección.

Vea también: **ip2long()**

openlog

(PHP 3, PHP 4 , PHP 5)

openlog -- abre una conexión con el logger del sistema

Descripción

int **openlog** (string ident, int option, int facility)

openlog() abre una conexión con el logger del sistema. La cadena *ident* se aÃ‚Â¤ade a cada mensaje. Los valores de *option* y *facility* se exponen en la siguiente sección. El uso de openlog() is opcional; Esta será llamada automaticamente por **syslog()** si fuera necesario, en este caso *ident* valdrá por defecto **FALSE**. Ver también **syslog()** y **closelog()**.

pfsockopen

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

pfsockopen -- Abre conexiones persistentes de dominio Internet o Unix.

Descripción

int **pfsockopen** (string hostname, int port [, int errno [, string errstr [, int timeout]]])

Esta función se comporta exactamente como **fsockopen()** con la diferencia que la conexion no se cierra después de que termine el script. Esta es la versión persistente de **fsockopen()**.

socket_get_status

socket_get_status -- Alias de **stream_get_meta_data()**

Descripción

Esta función es un alias de **stream_get_meta_data()**.

socket_set_blocking

socket_set_blocking -- Alias de **stream_set_blocking()**

Descripción

Esta función es un alias de **stream_set_blocking()**.

socket_set_timeout

socket_set_timeout -- Alias de **stream_set_timeout()**

Descripción

socket_set_timeout() es un alias de **stream_set_timeout()**.

syslog

(PHP 3, PHP 4 , PHP 5)

syslog -- genera un mensaje de sistema

Descripción

int **syslog** (int priority, string message)

syslog() genera un mensaje que será distribuido por el logger del sistema. *priority* es una combinación de la facility y el level, los valores se indicarán en la sección siguiente. El argumento restante es el mensaje a enviar, excepto que los dos caracteres *%m* sean reemplazados por la cadena de error (strerror) correspondiente al valor actual de errno.

Más información acerca de syslog se puede encontrar en las páginas del manual en equipos Unix.

En Windows NT, el servicio syslog es emulado usando el Log de Eventos.

LXXXIII. NIS funciona

NIS (anteriormente llamado Paginas Amarillas) permite la administracion de red de los archivos de administracion importantes (e.g.El archivo de contraseñas). Para mas informacion dirigirse a las paginas de ayuda de NIS y a la direccion. Introduccion a YP/NIS Hay tambien un libro llamado gestionando NFS Y NIS por Hal Stern.

Para obtener estas funciones de trabajo, usted tiene que configure PHP con -- *con- yp*.

Tabla de contenidos

yp_all

(PHP 4 >= 4.0.6, PHP 5)

yp_all -- Traverse the map and call a function on each entry

Description

void **yp_all** (string domain, string map, string callback)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

yp_cat

(PHP 4 >= 4.0.6, PHP 5)

yp_cat -- Return an array containing the entire map

Description

array **yp_cat** (string domain, string map)

yp_cat() returns all map entries as an array with the maps key values as array indices and the maps entries as array data.

yp_err_string

(PHP 4 >= 4.0.6, PHP 5)

yp_err_string -- devuelve el mensaje de error asociado con la operacion previa.Util que indica el problema exacto.

Descripcion

cadena **yp_err_string** (void)

yp_err_string() Retorna el mensaje de error asociado con la operacion previa.Util para indicar que salio mal exactamente.

Ejemplo 1. Ejemplo para errores de NIS

Vea tambien: yp errno

yp_errno

(PHP 4 >= 4.0.6, PHP 5)

yp_errno -- Retorna el codigo de error de la operacion previa.

Descripcion

int **yp_errno** (void)

yp_errno() retorna el codigo de error de la operacion previa.

Los errores posibles son:

1 args para funcionar son malos
2 fallo de RPC- dominio ha sido unbound
3 no puede unir a servidor en este dominio
4 ningun tal mapa en dominio de servidor
5 ninguna tal llave en
6 interno yp error de cliente o servidor
7 fallo de asignacion de recurso
8 ningunos más registros en base de datos de mapa
9 no puede comunicar wiTh portmapper
10 no puede comunicar con ypbind
11 no puede comunicar con ypserv
12 nombre de dominio local no conjunto
13 yp base de datos es malo
14 yp La version mismatch
15 violacion de acceso
16 base de datos ocupar
Ver tambien: yp_err_string

yp_first

yp_first -- devuelve la primera clave emparejada con el nombrado mapa.

Descripcion

```
string[] yp_first (cadena dominio, cadena mapa)
```

yp_first(nombre de la funcion)() Retorna la primera clave de valor pareada del mapa nombrado en el dominio, de otra manera FALSO.

Ejemplo 1. Ejemplo para el primer NIS

```
<?$entry= yp_first($dominio,passwd.byname");
            if (!$entry){"echo yp_errno().":".yp_err_string();}
           $key=key($entry);
      echo"Primera entrada en este mapa fue ".$key ."Y valor".$entry[$key];?>
```

Ver tambien: yp_get_default_domain yp_errno y yp_err_string

yp_get_default_domain

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

yp_get_default_domain -- Trae el valor por omision de dominios de maquina NIS.

Descripcion

int **yp_get_default_domain** (void)

yp_ get_default_domain() Retorna el valor por omision del dominio del nodo o FALSO. Puede ser

usado el parametro de dominio para sucesivas llamadas a NIS.

Un dominio de NIS puede ser descrito en un grupo de mapas NIS. Cada host necesita buscar uniones de informacion en un mismo dominio. Acudir a los documentos mencionados en el comienzo para mas informacion.

Ejemplo 1. Ejemplo para el dominio por omision

```
<?$domain = yp_ get_default_domain (); if (!$ domain) {"echo yp_errno()":" .yp_err_str
        echo "El valor por omision de dominio NIS es:  $ domain;?>
```

Ver tambien: yp_ errno (nombre de la funcion) y yp_err_string (nombre de la funcion)

yp_master

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

yp_master -- Returns the machine name of the master NIS server for a map

Description

string **yp_master** (string domain, string map)

yp_master() returns the machine name of the master NIS server for a map.

Ejemplo 1. Example for the NIS master

```
<?php
$number = yp_master($domain, $mapname);
echo "Master for this map is: " . $master;
?>
```

See also **yp_get_default_domain()**.

yp_match

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

yp_match -- Retorna la linea compañera (pareja).

Descripcion

cadena **yp_match** (cadena dominio, cadena mapa, cadena teclea)

yp_match(nombre de la funcion)() Retorna el valor asociado con la llave pasada fuera del mapa especificado o FALSO. esta llave tiene que ser exacta.

Ejemplo 1. Ejemplo para NIS parejo

En este caso esto puede ser: Joe:##joe:11111:100:joe usuario:/hogar/j/joe: User:/usr/local/bin/bash

Ver tambien: yp_get_default_domain yp_errno y yp_err_string

yp_next

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

yp_next -- Devuelve la siguiente clave tecleada en el nombre de mapa

Descripcion

string[] **yp_next** (cadena dominio, cadena mapa, cadena teclea)

yp_next() devuelve el siguiente par de valor tecleado en el mapa de nombres despues de la clave especificada o FALSO.

Ejemplo 1. Ejemplo para NIS siguiente

```
<?$entry=yp_next($dominio,"passwd. byname"."joe");
     if(!$entry){echo yp_errno()."":".yp_err_string();}
     $key=key($entry); echo "La siguiente entrada despues joe fue".$key."Y su valor".$e
```

Ver tambien: yp_get_default_domain yp_errno y yp_err_string

yp_order

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

yp_order -- Returns the order number for a map

Description

int **yp_order** (string domain, string map)

yp_order() returns the order number for a map or **FALSE**.

Ejemplo 1. Example for the NIS order

```
<?php
    $number = yp_order($domain, $mapname);
    echo "Order number for this map is: " . $number;
?>
```

See also **yp_get_default_domain()**.

LXXXIV. Lotus Notes Functions

Introducción

Aviso
Esta extensión es *EXPERIMENTAL*. Esto significa que el comportamiento de esta extensión, los nombre de sus funciones y en definitiva TODO lo documentado sobre esta extensión, puede cambiar en una futura versión de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Nota: This extension has been moved to the PECL repository and is no longer bundled with PHP as of PHP 5.0.0.

Tabla de contenidos

notes_body

(PHP 4 >= 4.0.5)

notes_body -- Open the message msg_number in the specified mailbox on the specified server (leave serv

Description

array **notes_body** (string server, string mailbox, int msg_number)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

notes_copy_db

(PHP 4 >= 4.0.5)

notes_copy_db -- Copy a Lotus Notes database

Description

string **notes_copy_db** (string from_database_name, string to_database_name)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

notes_create_db

(PHP 4 >= 4.0.5)

notes_create_db -- Create a Lotus Notes database

Description

bool **notes_create_db** (string database_name)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

notes_create_note

(PHP 4 >= 4.0.5)

notes_create_note -- Create a note using form form_name

Description

string **notes_create_note** (string database_name, string form_name)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

notes_drop_db

(PHP 4 >= 4.0.5)

notes_drop_db -- Drop a Lotus Notes database

Description

bool **notes_drop_db** (string database_name)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

notes_find_note

(PHP 4 >= 4.0.5)

notes_find_note -- Returns a note id found in database_name

Description

bool **notes_find_note** (string database_name, string name [, string type])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

notes_header_info

(PHP 4 >= 4.0.5)

notes_header_info -- Open the message msg_number in the specified mailbox on the specified server (leave serv

Description

object **notes_header_info** (string server, string mailbox, int msg_number)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

notes_list_msgs

(PHP 4 >= 4.0.5)

notes_list_msgs -- Returns the notes from a selected database_name

Description

bool **notes_list_msgs** (string db)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

notes_mark_read

(PHP 4 >= 4.0.5)

notes_mark_read -- Mark a note_id as read for the User user_name

Description

string **notes_mark_read** (string database_name, string user_name, string note_id)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

notes_mark_unread

(PHP 4 >= 4.0.5)

notes_mark_unread -- Mark a note_id as unread for the User user_name

Description

string **notes_mark_unread** (string database_name, string user_name, string note_id)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

notes_nav_create

(PHP 4 >= 4.0.5)

notes_nav_create -- Create a navigator name, in database_name

Description

bool **notes_nav_create** (string database_name, string name)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

notes_search

(PHP 4 >= 4.0.5)

notes_search -- Find notes that match keywords in database_name

Description

string **notes_search** (string database_name, string keywords)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

notes_unread

(PHP 4 >= 4.0.5)

notes_unread -- Returns the unread note id's for the current User user_name

Description

string **notes_unread** (string database_name, string user_name)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

notes_version

(PHP 4 >= 4.0.5)

notes_version -- Get the version Lotus Notes

Description

string **notes_version** (string database_name)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

LXXXV. NSAPI-specific Functions

Introducción

These functions are only available when running PHP as a NSAPI module in Netscape/iPlanet/SunONE webservers.

Instalación

For PHP installation on Netscape/iPlanet/SunONE webservers see the NSAPI section (UNIX, Windows) in the installation chapter.

Configuración en tiempo de ejecución

The behaviour of the NSAPI PHP module is affected by settings in `php.ini`. Configuration settings from `php.ini` may be overridden by additional parameters to the *php4_execute* call in `obj.conf`

Tabla 1. NSAPI configuration options

Name	Default	Changeable
nsapi.read_timeout	60	PHP_INI_ALL

For further details and definitions of the PHP_INI_* constants, see the Apéndice H.

A continuación se presenta una corta explicación de las directivas de configuración.

nsapi.read_timeout **integer**

Sets the time in seconds the plugin is waiting for POST data from the client.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Esta extensión no tiene ninguna constante definida.

Ver también

NSAPI implements a subset of the functions from the Apache module for maximum compatibility.

Tabla 2. Apache functions implemented by NSAPI

Apache function (only as alias)	NSAPI function	Description
apache_request_headers()	nsapi_request_headers()	Fetch all HTTP request headers
apache_response_headers()	nsapi_response_headers()	Fetch all HTTP response headers
getallheaders()	nsapi_request_headers()	Fetch all HTTP request headers
virtual()	nsapi_virtual()	Make NSAPI sub-request

Tabla de contenidos

nsapi_request_headers

(PHP 4 >= 4.3.3, PHP 5)

nsapi_request_headers -- Fetch all HTTP request headers

Descripción

array **nsapi_request_headers** (void)

nsapi_request_headers() gets all the HTTP headers in the current request. This is only supported when PHP runs as a NSAPI module.

> **Nota:** Prior to PHP 4.3.3, **getallheaders()** was only available for the Apache servers. After PHP 4.3.3, **getallheaders()** is an alias for **nsapi_request_headers()** if you use the NSAPI module.

> **Nota:** You can also get at the value of the common CGI variables by reading them from the *$_SERVER* superglobal, which works whether or not you are using PHP as a NSAPI module.

Valores retornados

Returns an associative array with all the HTTP headers.

Ejemplos

Ejemplo 1. nsapi_request_headers() example

```
<?php
$headers = nsapi_request_headers();

foreach ($headers as $header => $value) {
  echo "$header: $value <br />\n";
}
?>
```

nsapi_response_headers

(PHP 4 >= 4.3.3, PHP 5)

nsapi_response_headers -- Fetch all HTTP response headers

Descripción

array **nsapi_response_headers** (void)

Gets all the NSAPI response headers.

Valores retornados

Returns an associative array with all the NSAPI response headers.

Ver también

nsapi_request_headers()
headers_sent()

nsapi_virtual

(PHP 4 >= 4.3.3, PHP 5)

nsapi_virtual -- Perform an NSAPI sub-request

Descripción

bool **nsapi_virtual** (string uri)

nsapi_virtual() is an NSAPI-specific function which is ñalent to <!--#include virtual...--> in SSI (`.shtml` files). It does an NSAPI sub-request. It is useful for including CGI scripts or .shtml files, or anything else that you'd parse through webserver.

To run the sub-request, all buffers are terminated and flushed to the browser, pending headers are sent too.

You cannot make recursive requests with this function to other PHP scripts. If you want to include PHP scripts, use **include()** or **require()**.

> **Nota:** This function depends on a undocumented feature of the Netscape/iPlanet/SunONE webservers. Use **phpinfo()** to determine if it is available. In the Unix environment it should always work, in windows it depends on the name of a `ns-httpdXX.dll` file.
>
> Read the note about subrequests in the NSAPI section (UNIX, Windows) if you experience this problem.

Lista de parámetros

uri

> The URI of the script.

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

LXXXVI. Object Aggregation/Composition Functions

Aviso
Esta extensión es *EXPERIMENTAL*. Esto significa que el comportamiento de esta extensión, los nombre de sus funciones y en definitiva TODO lo documentado sobre esta extensión, puede cambiar en una futura versión de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Introducción

In Object Oriented Programming, it is common to see the composition of simple classes (and/or instances) into a more complex one. This is a flexible strategy for building complicated objects and object hierarchies and can function as a dynamic alternative to multiple inheritance. There are two ways to perform class (and/or object) composition depending on the relationship between the composed elements: *Association* and *Aggregation*.

An *Association* is a composition of independently constructed and externally visible parts. When we associate classes or objects, each one keeps a reference to the ones it is associated with. When we associate classes statically, one class will contain a reference to an instance of the other class. For example:
Ejemplo 1. Class association

```
<?php
class DateTime {

   function DateTime()
   {
       // empty constructor
   }

   function now()
   {
       return date("Y-m-d H:i:s");
   }
}

class Report {
   var $_dt;
   // more properties ...

   function Report()
   {
       $this->_dt = new DateTime();
       // initialization code ...
   }

   function generateReport()
   {
       $dateTime = $this->_dt->now();
       // more code ...
   }

   // more methods ...
}

$rep = new Report();
?>
```

We can also associate instances at runtime by passing a reference in a constructor (or any other method), which allow us to dynamically change the association relationship between objects. We will modify the example above to illustrate this point:

Ejemplo 2. Object association

```
<?php
class DateTime {
   // same as previous example
}

class DateTimePlus {
   var $_format;

   function DateTimePlus($format="Y-m-d H:i:s")
   {
       $this->_format = $format;
   }

   function now()
   {
       return date($this->_format);
   }
}

class Report {
   var $_dt;    // we'll keep the reference to DateTime here
   // more properties ...

   function Report()
   {
       // do some initialization
   }

   function setDateTime(&$dt)
   {
       $this->_dt =& $dt;
   }

   function generateReport()
   {
       $dateTime = $this->_dt->now();
       // more code ...
   }

   // more methods ...
}

$rep = new Report();
$dt = new DateTime();
$dtp = new DateTimePlus("l, F j, Y (h:i:s a, T)");

// generate report with simple date for web display
$rep->setDateTime(&$dt);
echo $rep->generateReport();

// later on in the code ...

// generate report with fancy date
$rep->setDateTime(&$dtp);
$output = $rep->generateReport();
// save $output in database
// ... etc ...
?>
```

Aggregation, on the other hand, implies encapsulation (hidding) of the parts of the composition. We can aggregate classes by using a (static) inner class (PHP does not yet support inner classes), in this case the aggregated class definition is not accessible, except through the class that contains it. The aggregation of instances (object aggregation) involves the dynamic creation of subobjects inside an object, in the process, expanding the properties and methods of that object.

Object aggregation is a natural way of representing a whole-part relationship, (for example, molecules are aggregates of atoms), or can be used to obtain an effect ñalent to multiple inheritance, without having to permanently bind a subclass to two or more parent classes and their interfaces. In

fact object aggregation can be more flexible, in which we can select what methods or properties to "inherit" in the aggregated object.

Ejemplos

We define 3 classes, each implementing a different storage method:

Ejemplo 3. storage_classes.inc

```
<?php
class FileStorage {
    var $data;

    function FileStorage($data)
    {
        $this->data = $data;
    }

    function write($name)
    {
        $fp = fopen(name, "w");
        fwrite($fp, $this->data);
        fclose($data);
    }
}

class WDDXStorage {
    var $data;
    var $version = "1.0";
    var $_id; // "private" variable

    function WDDXStorage($data)
    {
        $this->data = $data;
        $this->_id = $this->_genID();
    }

    function store()
    {
        if ($this->_id) {
            $pid = wddx_packet_start($this->_id);
            wddx_add_vars($pid, "this->data");
            $packet = wddx_packet_end($pid);
        } else {
            $packet = wddx_serialize_value($this->data);
        }
        $dbh = dba_open("varstore", "w", "gdbm");
        dba_insert(md5(uniqid("", true)), $packet, $dbh);
        dba_close($dbh);
    }

    // a private method
    function _genID()
    {
        return md5(uniqid(rand(), true));
    }
}

class DBStorage {
    var $data;
    var $dbtype = "mysql";

    function DBStorage($data)
    {
        $this->data = $data;
    }

    function save()
    {
        $dbh = mysql_connect();
        mysql_select_db("storage", $dbh);
        $serdata = serialize($this->data);
        mysql_query("insert into vars ('$serdata',now())", $dbh);
        mysql_close($dbh);
    }
}

?>
```

We then instantiate a couple of objects from the defined classes, and perform some aggregations

and deaggregations, printing some object information along the way:

Ejemplo 4. test_aggregation.php

```
<?php
include "storageclasses.inc";

// some utilty functions

function p_arr($arr)
{
    foreach ($arr as $k => $v)
        $out[] = "\t$k => $v";
    return implode("\n", $out);
}

function object_info($obj)
{
    $out[] = "Class: " . get_class($obj);
    foreach (get_object_vars($obj) as $var=>$val) {
        if (is_array($val)) {
            $out[] = "property: $var (array)\n" . p_arr($val);
        } else {
            $out[] = "property: $var = $val";
        }
    }
    foreach (get_class_methods($obj) as $method) {
        $out[] = "method: $method";
    }
    return implode("\n", $out);
}


$data = array(M_PI, "kludge != cruft");

// we create some basic objects
$fs = new FileStorage($data);
$ws = new WDDXStorage($data);

// print information on the objects
echo "\$fs object\n";
echo object_info($fs) . "\n";
echo "\n\$ws object\n";
echo object_info($ws) . "\n";

// do some aggregation

echo "\nLet's aggregate \$fs to the WDDXStorage class\n";
aggregate($fs, "WDDXStorage");
echo "\$fs object\n";
echo object_info($fs) . "\n";

echo "\nNow let us aggregate it to the DBStorage class\n";
aggregate($fs, "DBStorage");
echo "\$fs object\n";
echo object_info($fs) . "\n";

echo "\nAnd finally deaggregate WDDXStorage\n";
deaggregate($fs, "WDDXStorage");
echo "\$fs object\n";
echo object_info($fs) . "\n";

?>
```

We will now consider the output to understand some of the side-effects and limitation of object aggregation in PHP. First, the newly created *$fs* and *$ws* objects give the expected output (according to their respective class declaration). Note that for the purposes of object aggregation, *private elements of a class/object begin with an underscore character ("_")*, even though there is not real distinction between public and private class/object elements in PHP.

```
$fs object
Class: filestorage
property: data (array)
    0 => 3.1415926535898
    1 => kludge != cruft
method: filestorage
method: write

$ws object
Class: wddxstorage
property: data (array)
    0 => 3.1415926535898
    1 => kludge != cruft
property: version = 1.0
property: _id = ID::9bb2b640764d4370eb04808af8b076a5
method: wddxstorage
method: store
method: _genid
```

We then aggregate *$fs* with the **WDDXStorage** class, and print out the object information. We can see now that even though nominally the *$fs* object is still of **FileStorage**, it now has the property *$version*, and the method **store()**, both defined in **WDDXStorage**. One important thing to note is that it has not aggregated the private elements defined in the class, which are present in the *$ws* object. Also absent is the constructor from **WDDXStorage**, which will not be logical to aggegate.

```
Let's aggregate $fs to the WDDXStorage class
$fs object
Class: filestorage
property: data (array)
    0 => 3.1415926535898
    1 => kludge != cruft
property: version = 1.0
method: filestorage
method: write
method: store
```

The process of aggregation is cumulative, so when we aggregate *$fs* with the class **DBStorage**, generating an object that can use the storage methods of all the defined classes.

```
Now let us aggregate it to the DBStorage class
$fs object
Class: filestorage
property: data (array)
    0 => 3.1415926535898
    1 => kludge != cruft
property: version = 1.0
property: dbtype = mysql
method: filestorage
method: write
method: store
method: save
```

Finally, the same way we aggregated properties and methods dynamically, we can also deaggregate them from the object. So, if we deaggregate the class **WDDXStorage** from *$fs*, we will obtain:

```
And deaggregate the WDDXStorage methods and properties
$fs object
Class: filestorage
property: data (array)
    0 => 3.1415926535898
    1 => kludge != cruft
property: dbtype = mysql
method: filestorage
method: write
method: save
```

One point that we have not mentioned above, is that the process of aggregation will not override

existing properties or methods in the objects. For example, the class **FileStorage** defines a *$data* property, and the class **WDDXStorage** also defines a similar property which will not override the one in the object acquired during instantiation from the class **FileStorage**.

Tabla de contenidos

aggregate_info

(no version information, might be only in CVS)

aggregate_info -- Returns an associative array of the methods and properties from each class that has been aggregated to the object

Description

array **aggregate_info** (object object)

Will return the aggregation information for a particular object as an associative array of arrays of methods and properties. The key for the main array is the name of the aggregated class.

For example the code below
Ejemplo 1. Using aggregate_info()

```
<?php

class Slicer {
    var $vegetable;

    function Slicer($vegetable)
    {
        $this->vegetable = $vegetable;
    }

    function slice_it($num_cuts)
    {
        echo "Doing some simple slicing\n";
        for ($i=0; $i < $num_cuts; $i++) {
            // do some slicing
        }
    }
}

class Dicer {
    var $vegetable;
    var $rotation_angle = 90;   // degrees

    function Dicer($vegetable)
    {
        $this->vegetable = $vegetable;
    }

    function dice_it($num_cuts)
    {
        echo "Cutting in one direction\n";
        for ($i=0; $i < $num_cuts; $i++) {
            // do some cutting
        }
        $this->rotate($this->rotation_angle);
        echo "Cutting in a second direction\n";
        for ($i=0; $i < $num_cuts; $i++) {
            // do some more cutting
        }
    }

    function rotate($deg)
    {
        echo "Now rotating {$this->vegetable} {$deg} degrees\n";
    }

    function _secret_super_dicing($num_cuts)
    {
        // so secret we cannot show you ;-)
    }
}

$obj = new Slicer('onion');
aggregate($obj, 'Dicer');
print_r(aggregate_info($obj));
?>
```

Will produce the output

```
Array
(
    [dicer] => Array
        (
            [methods] => Array
                (
                    [0] => dice_it
                    [1] => rotate
                )

            [properties] => Array
                (
                    [0] => rotation_angle
                )

        )

)
```

As you can see, all properties and methods of the **Dicer** class have been aggregated into our new object, with the exception of the class constructor and the method `_secret_super_dicing`

See also **aggregate()**, **aggregate_methods()**, **aggregate_methods_by_list()**, **aggregate_methods_by_regexp()**, **aggregate_properties()**, **aggregate_properties_by_list()**, **aggregate_properties_by_regexp()**, **deaggregate()**

aggregate_methods_by_list

(PHP 4 >= 4.2.0)

aggregate_methods_by_list -- Selective dynamic class methods aggregation to an object

Description

void **aggregate_methods_by_list** (object object, string class_name, array methods_list [, bool exclude])

Aggregates methods from a class to an existing object using a list of method names. The optional parameter *exclude* is used to decide whether the list contains the names of methods to include in the aggregation (i.e. *exclude* is **FALSE**, which is the default value), or to exclude from the aggregation (*exclude* is **TRUE**).

The class constructor or methods whose names start with an underscore character (_), which are considered private to the aggregated class, are always excluded.

See also **aggregate()**, **aggregate_info()**, **aggregate_methods()**, **aggregate_methods_by_regexp()**, **aggregate_properties()**, **aggregate_properties_by_list()**, **aggregate_properties_by_regexp()**, **deaggregate()**

aggregate_methods_by_regexp

(PHP 4 >= 4.2.0)

aggregate_methods_by_regexp -- Selective class methods aggregation to an object using a regular expression

Description

void **aggregate_methods_by_regexp** (object object, string class_name, string regexp [, bool exclude])

Aggregates methods from a class to an existing object using a regular expression to match method names. The optional parameter *exclude* is used to decide whether the regular expression will select the names of methods to include in the aggregation (i.e. *exclude* is **FALSE**, which is the default value), or to exclude from the aggregation (*exclude* is **TRUE**).

The class constructor or methods whose names start with an underscore character (_), which are considered private to the aggregated class, are always excluded.

See also **aggregate()**, **aggregate_info()**, **aggregate_methods()**, **aggregate_methods_by_list()**, **aggregate_properties()**, **aggregate_properties_by_list()**, **aggregate_properties_by_regexp()**, **deaggregate()**

aggregate_methods

(PHP 4 >= 4.2.0)

aggregate_methods -- Dynamic class and object aggregation of methods

Description

void **aggregate_methods** (object object, string class_name)

Aggregates all methods defined in a class to an existing object, except for the class constructor, or methods whose names start with an underscore character (_) which are considered private to the aggregated class.

See also **aggregate()**, **aggregate_info()**, **aggregate_methods_by_list()**, **aggregate_methods_by_regexp()**, **aggregate_properties()**, **aggregate_properties_by_list()**, **aggregate_properties_by_regexp()**, **deaggregate()**

aggregate_properties_by_list

(PHP 4 >= 4.2.0)

aggregate_properties_by_list -- Selective dynamic class properties aggregation to an object

Description

void **aggregate_properties_by_list** (object object, string class_name, array properties_list [, bool exclude])

Aggregates properties from a class to an existing object using a list of property names. The optional parameter *exclude* is used to decide whether the list contains the names of class properties to include in the aggregation (i.e. *exclude* is **FALSE**, which is the default value), or to exclude from the aggregation (*exclude* is **TRUE**).

The properties whose names start with an underscore character (_), which are considered private to the aggregated class, are always excluded.

See also **aggregate()**, **aggregate_methods()**, **aggregate_methods_by_list()**, **aggregate_methods_by_regexp()**, **aggregate_properties()**, **aggregate_properties_by_regexp()**, **aggregate_info()**, **deaggregate()**

aggregate_properties_by_regexp

(PHP 4 >= 4.2.0)

aggregate_properties_by_regexp -- Selective class properties aggregation to an object using a regular expression

Description

void **aggregate_properties_by_regexp** (object object, string class_name, string regexp [, bool exclude])

Aggregates properties from a class to an existing object using a regular expression to match their names. The optional parameter *exclude* is used to decide whether the regular expression will select the names of class properties to include in the aggregation (i.e. *exclude* is **FALSE**, which is the default value), or to exclude from the aggregation (*exclude* is **TRUE**).

The properties whose names start with an underscore character (_), which are considered private to the aggregated class, are always excluded.

See also **aggregate()**, **aggregate_methods()**, **aggregate_methods_by_list()**, **aggregate_methods_by_regexp()**, **aggregate_properties()**, **aggregate_properties_by_list()**, **aggregate_info()**, **deaggregate()**

aggregate_properties

(PHP 4 >= 4.2.0)

aggregate_properties -- Dynamic aggregation of class properties to an object

Description

void **aggregate_properties** (object object, string class_name)

Aggregates all properties defined in a class to an existing object, except for properties whose names start with an underscore character (_) which are considered private to the aggregated class.

See also **aggregate()**, **aggregate_methods()**, **aggregate_methods_by_list()**, **aggregate_methods_by_regexp()**, **aggregate_properties_by_list()**, **aggregate_properties_by_regexp()**, **aggregate_info()**, **deaggregate()**

aggregate

(PHP 4 >= 4.2.0)

aggregate -- Dynamic class and object aggregation of methods and properties

Description

void **aggregate** (object object, string class_name)

Aggregates methods and properties defined in a class to an existing object. Methods and properties with names starting with an underscore character (_) are considered private to the aggregated class and are not used, constructors are also excluded from the aggregation procedure.

See also **aggregate_info()**, **aggregate_methods()**, **aggregate_methods_by_list()**, **aggregate_methods_by_regexp()**, **aggregate_properties()**, **aggregate_properties_by_list()**, **aggregate_properties_by_regexp()**, **deaggregate()**

aggregation_info

aggregation_info -- Alias of **aggregate_info()**

Description

This function is an alias of **aggregate_info()**.

deaggregate

(PHP 4 >= 4.2.0)

deaggregate -- Removes the aggregated methods and properties from an object

Description

void **deaggregate** (object object [, string class_name])

Removes the methods and properties from classes that were aggregated to an object. If the optional *class_name* parameters is passed, only those methods and properties defined in that class are removed, otherwise all aggregated methods and properties are eliminated.

See also **aggregate()**, **aggregate_methods()**, **aggregate_methods_by_list()**, **aggregate_methods_by_regexp()**, **aggregate_properties()**, **aggregate_properties_by_list()**, **aggregate_properties_by_regexp()**, **aggregate_info()**

LXXXVII. Funciones de Oracle 8

Introducción

Estas funciones le permiten acceder a bases de datos Oracle9, Oracle8 y Oracle7. Usan la Interfaz de Llamadas Oracle (OCI por sus siglas en Inglés).

Esta extensión es más flexible que la extensión antigua de Oracle. Ésta soporta la vinculación de variables PHP globales y locales con recipientes Oracle, tiene soporte completo LOB, FILE y ROWID, y le permite usar variables de definición entregadas por el usuario. Es recomendable usar esta extensión en lugar de la extensión antigua de Oracle siempre que sea posible.

Requirimientos

Es necesario contar con las bibliotecas de cliente Oracle para usar esta extensión. Los usuarios de Windows necesitan por lo menos la versión 8.1 de Oracle para usar `php_oci8.dll`.

Antes de usar esta extensión, asegúrese de haber configurado apropiadamente sus variables de entorno Oracle para que correspondan con el usuario de Oracle, así como su usuario del demonio web. Las variables que necesita definir son las siguientes:

- ORACLE_HOME
- ORACLE_SID
- LD_PRELOAD
- LD_LIBRARY_PATH
- NLS_LANG
- ORA_NLS33

Después de definir las variables de entorno para su usuario de servidor web, asegúrese también de agregar al usuario del servidor web (nobody, www) al grupo oracle.

> **Si su servidor web no arranca, o falla al arrancar:** Verifique que Apache esté enlazado con la biblioteca pthread:

```
# ldd /www/apache/bin/httpd
    libpthread.so.0 => /lib/libpthread.so.0 (0x4001c000)
    libm.so.6 => /lib/libm.so.6 (0x4002f000)
    libcrypt.so.1 => /lib/libcrypt.so.1 (0x4004c000)
    libdl.so.2 => /lib/libdl.so.2 (0x4007a000)
    libc.so.6 => /lib/libc.so.6 (0x4007e000)
    /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

Si libpthread no se encuentra en la lista, necesita re-instalar Apache:

```
# cd /usr/src/apache_1.3.xx
# make clean
# LIBS=-lpthread ./config.status
# make
# make install
```

Por favor note que en algunos sistemas, como UnixWare, se trata de libthread en lugar de libpthread. PHP y Apache deben configurarse con EXTRA_LIBS=-lthread.

Instalación

You have to compile PHP with the option *--with-oci8[=DIR]*, where DIR defaults to your environment variable ORACLE_HOME.

If you're using Oracle Instant Client, you need to build PHP with the option *--with-oci8-instant-client[=DIR]*. Note that Oracle Instant Client support first appeared in versions 4.3.11 and 5.0.4.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

`OCI_DEFAULT` (**integer**)

Statement execution mode. Statement is not committed automatically when using this mode.

`OCI_DESCRIBE_ONLY` (**integer**)

Statement execution mode. Use this mode if you don't want to really execute query, but only get select-list description.

`OCI_COMMIT_ON_SUCCESS` (**integer**)

Statement execution mode. Statement is automatically committed after **oci_execute()** call.

`OCI_EXACT_FETCH` (**integer**)

Statement fetch mode. Used when the application knows in advance exactly how many rows it will be fetching. This mode turns prefetching off for Oracle release 8 or later mode. Cursor is cancelled after the desired rows are fetched and may result in reduced server-side resource

usage.

OCI_SYSDATE (integer)

OCI_B_BFILE (integer)

Used with oci_bind_by_name() when binding BFILEs.

OCI_B_CFILEE (integer)

Used with oci_bind_by_name() when binding CFILEs.

OCI_B_CLOB (integer)

Used with oci_bind_by_name() when binding CLOBs.

OCI_B_BLOB (integer)

Used with oci_bind_by_name() when binding BLOBs.

OCI_B_ROWID (integer)

Used with oci_bind_by_name() when binding ROWIDs.

OCI_B_CURSOR (integer)

Used with oci_bind_by_name() when binding cursors, previously allocated with oci_new_descriptor().

OCI_B_NTY (integer)

Used with oci_bind_by_name() when binding named data types.

OCI_B_BIN (integer)

SQLT_BFILEE (integer)

The same as **OCI_B_BFILE**.

SQLT_CFILEE (integer)

The same as **OCI_B_CFILEE**.

SQLT_CLOB (integer)

The same as **OCI_B_CLOB**.

SQLT_BLOB (integer)

The same as **OCI_B_BLOB**.

SQLT_RDD (integer)

The same as **OCI_B_ROWID**.

SQLT_NTY (integer)

The same as **OCI_B_NTY**.

OCI_FETCHSTATEMENT_BY_COLUMN (integer)

Default mode of oci_fetch_all().

OCI_FETCHSTATEMENT_BY_ROW (integer)

Alternative mode of oci_fetch_all().

OCI_ASSOC (integer)

Used with oci_fetch_all() and oci_fetch_array() to get an associative array as a result.

OCI_NUM (integer)

Used with oci_fetch_all() and oci_fetch_array() to get an enumerated array as a result.

OCI_BOTH (integer)

Used with oci_fetch_all() and oci_fetch_array() to get an array with both associative and number indices.

OCI_RETURN_NULLS (integer)

Used with oci_fetch_array() to get empty array elements if field's value is **NULL**.

OCI_RETURN_LOBS (integer)

Used with oci_fetch_array() to get value of LOB instead of the descriptor.

OCI_DTYPE_FILE (integer)

This flag tells oci_new_descriptor() to initialize new FILE descriptor.

OCI_DTYPE_LOB (integer)

This flag tells oci_new_descriptor() to initialize new LOB descriptor.

OCI_DTYPE_ROWID (integer)

This flag tells oci_new_descriptor() to initialize new ROWID descriptor.

OCI_D_FILE (integer)

The same as **OCI_DTYPE_FILE**.

OCI_D_LOB (integer)

The same as **OCI_DTYPE_LOB**.

OCI_D_ROWID (**integer**)

The same as **OCI_DTYPE_ROWID**.

Ejemplos

Ejemplo 1. Consejos OCI

```
<?php
// por sergo arroba bacup punto ru

// Use la opcion OCI_DEFAULT en el comando de ejecucion para aplazar
// la ejecucion
OCIExecute($sentencia, OCI_DEFAULT);

// para recuperar datos use (despues de fetch):

$resultado = OCIResult($sentencia, $n);
if (is_object($result)) $resultado = $resultado->load();

// Para una sentencia INSERT o UPDATE use:

$sql = "insert into tabla (campo1, campo2) values (campo1 = 'valor',
 campo2 = empty_clob()) returning campo2 into :campo2";
OCIParse($con, $sql);
$clob = OCINewDescriptor($con, OCI_D_LOB);
OCIBindByName($sentencia, ":campo2", &$clob, -1, OCI_B_CLOB);
OCIExecute($sentencia, OCI_DEFAULT);
$clob->save("cualquier texto");
OCICommit($con);

?>
```

Es posible acceder fácilmente a procedimientos almacenados en la misma forma en que lo haría desde la línea de comandos.

Ejemplo 2. Uso de Procedimientos Almacenados

```
<?php
// por webmaster arroba remoterealty punto com
$sth = OCIParse($dbh, "begin proc_nueva_dir( :id_direccion, '$nombre',
 '$apellido', '$companyia', '$dir1', '$dir2', '$ciudad', '$estado',
 '$cod_postal', '$pais', :cod_error );end;");

// Esto llama al procedimiento almacenado proc_nueva_dir, en donde
// :id_direccion es una variable de entrada/salida y :cod_error es una
// variable de salida.
// Entonces crea la vinculacion:

   OCIBindByName($sth, ":id_direccion", $id_direccion, 10);
   OCIBindByName($sth, ":cod_error", $cod_error, 10);
   OCIExecute($sth);

?>
```

Tabla de contenidos

OCI-Collection->free -- Frees resources associated with collection object
OCI-Collection->max -- Gets the maximum number of elements in the collection
OCI-Collection->size -- Returns size of the collection
OCI-Collection->trim -- Trims elements from the end of the collection
oci_commit -- Commits outstanding statements
oci_connect -- Establishes a connection to Oracle server
oci_define_by_name -- Uses a PHP variable for the define-step during a SELECT
oci_error -- Returns the last error found
oci_execute -- Executes a statement
oci_fetch_all -- Fetches all rows of result data into an array
oci_fetch_array -- Returns the next row from the result data as an associative or numeric array, or both
oci_fetch_assoc -- Returns the next row from the result data as an associative array
oci_fetch_object -- Returns the next row from the result data as an object
oci_fetch_row -- Returns the next row from the result data as a numeric array
oci_fetch -- Fetches the next row into result-buffer
oci_field_is_null -- Checks if the field is **NULL**
oci_field_name -- Returns the name of a field from the statement
oci_field_precision -- Tell the precision of a field
oci_field_scale -- Tell the scale of the field
oci_field_size -- Returns field's size
oci_field_type_raw -- Tell the raw Oracle data type of the field
oci_field_type -- Returns field's data type
descriptor->free -- Frees resources associated with descriptor
oci_free_statement -- Frees all resources associated with statement or cursor
oci_internal_debug -- Enables or disables internal debug output
lob->append -- Appends data from the large object to another large object
lob->close -- Closes LOB descriptor
oci_lob_copy -- Copies large object
lob->eof -- Tests for end-of-file on a large object's descriptor
lob->erase -- Erases a specified portion of the internal LOB data
lob->export -- Exports LOB's contents to a file
lob->flush -- Flushes/writes buffer of the LOB to the server
lob->import -- Imports file data to the LOB
oci_lob_is_equal -- Compares two LOB/FILE locators for equality
lob->load -- Returns large object's contents
lob->read -- Reads part of large object
lob->rewind -- Moves the internal pointer to the beginning of the large object
lob->save -- Saves data to the large object
lob->seek -- Sets the internal pointer of the large object
lob->size -- Returns size of large object
lob->tell -- Returns current position of internal pointer of large object
lob->truncate -- Truncates large object
lob->writeTemporary -- Writes temporary large object
lob->write -- Writes data to the large object
oci_new_collection -- Allocates new collection object
oci_new_connect -- Establishes a new connection to the Oracle server
oci_new_cursor -- Allocates and returns a new cursor (statement handle)
oci_new_descriptor -- Initializes a new empty LOB or FILE descriptor
oci_num_fields -- Returns the number of result columns in a statement
oci_num_rows -- Returns number of rows affected during statement execution
oci_parse -- Prepares Oracle statement for execution
oci_password_change -- Changes password of Oracle's user
oci_pconnect -- Connect to an Oracle database using a persistent connection
oci_result -- Returns field's value from the fetched row

oci_rollback -- Rolls back outstanding transaction
oci_server_version -- Returns server version
oci_set_prefetch -- Sets number of rows to be prefetched
oci_statement_type -- Returns the type of an OCI statement
OCIBindByName -- Enlaza una variable PHP a un Placeholder de Oracle
ocicancel -- Cancel reading from cursor
ocicloselob -- Closes lob descriptor
ocicollappend -- Append an object to the collection
ocicollassign -- Assign a collection from another existing collection
ocicollassignelem -- Assign element val to collection at index ndx
ocicollgetelem -- Retrieve the value at collection index ndx
ocicollmax -- Gets the maximum number of elements in the collection
ocicollsize -- Return the size of a collection
ocicolltrim -- Trim num elements from the end of a collection
OCIColumnIsNULL -- comprueba si una una columna es **NULL**
OCIColumnName -- Devuelve el nombre de una columna.
ocicolumnprecision -- Tell the precision of a column
ocicolumnscale -- Tell the scale of a column
OCIColumnSize -- devuelve el tamaño de la columna
OCIColumnType -- Devuelve el tipo de dato de una columna.
ocicolumntyperaw -- Tell the raw oracle data type of a column
OCICommit -- Confirma transacciones pendientes
OCIDefineByName -- Usa una variable de PHP para el define-step durante una sentencia SELECT
OCIError -- Devuelve el último error de stmt|conn|global. Si no ocurre ningún error devuelve falso.
OCIExecute -- Ejecuta una sentencia
OCIFetch -- Busca la siguiente fila en el result-buffer
OCIFetchInto -- Busca la siguiente fila dentro del result-array
OCIFetchStatement -- Busca todas la filas de un resultset dentro de un array.
ocifreecollection -- Deletes collection object
OCIFreeCursor -- Libera todos los recursos asociados con cursor.
ocifreedesc -- Deletes a large object descriptor
OCIFreeStatement -- Libera todos los recursos asociados con una sentencia.
lob->getBuffering -- Returns current state of buffering for large object
OCIInternalDebug -- Habilita o deshabilita la salida del depurador interno. Por defecto este está deshabiltado
ociloadlob -- Loads a large object
OCILogOff -- Termina la conexion con Oracle
ocilogon -- Establece una conexión con Oracle
ocinewcollection -- Initialize a new collection
OCINewCursor -- devuelve un cursor nuevo (Statement-Handle) - use esto para enlazar ref-cursors!
OCINewDescriptor -- Inicializa un nuevo descriptor vacío LOB/FILE (LOB por defecto)
ocinlogon -- Establece una nueva conexión con Oracle
OCINumCols -- Devuelve el número de columnas resultantes en una sentencia
OCIParse -- Analiza una consulta y devuelve una sentencia
ociplogon -- Conectarse con una base de datos Oracle usando una conexión persistente
OCIResult -- Devuelve el valor de una columna en la fila buscada
OCIRollback -- Restablece todas las transaciones sin confirmar
OCIRowCount -- Obtiene el número de filas afectadas
ocisavelob -- Saves a large object
ocisavelobfile -- Saves a large object file
OCIServerVersion -- Devuelve una cadena conteniendo información a cerca de la version del servidor.
lob->setBuffering -- Changes current state of buffering for large object
ocisetprefetch -- Sets number of rows to be prefetched
OCIStatementType -- Devuelve el tipo de una sentencia OCI.

ociwritelobtofile -- Saves a large object file
ociwritetemporarylob -- Writes temporary blob

oci_bind_by_name

(PHP 5)

oci_bind_by_name -- Binds the PHP variable to the Oracle placeholder

Description

bool **oci_bind_by_name** (resource stmt, string ph_name, mixed &variable [, int maxlength [, int type]])

oci_bind_by_name() binds the PHP variable *variable* to the Oracle placeholder *ph_name*. Whether it will be used for input or output will be determined at run-time and the necessary storage space will be allocated. The *length* parameter sets the maximum length for the bind. If you set *length* to -1 **oci_bind_by_name()** will use the current length of *variable* to set the maximum length.

If you need to bind an abstract datatype (LOB/ROWID/BFILE) you need to allocate it first using the **oci_new_descriptor()** function. The *length* is not used for abstract datatypes and should be set to -1. The *type* parameter tells Oracle which descriptor is used. Possible values are:

- **OCI_B_FILE** - for BFILEs;
- **OCI_B_CFILE** - for CFILEs;
- **OCI_B_CLOB** - for CLOBs;
- **OCI_B_BLOB** - for BLOBs;
- **OCI_B_ROWID** - for ROWIDs;
- **OCI_B_NTY** - for named datatypes;
- **OCI_B_CURSOR** - for cursors, that were created before with **oci_new_cursor()**.

Ejemplo 1. oci_bind_by_name()example

```
<?php
/* oci_bind_by_name example thies at thieso dot net (980221)
  inserts 3 records into emp, and uses the ROWID for updating the
  records just after the insert.
*/

$conn = oci_connect("scott", "tiger");

$stmt = oci_parse($conn, "
                          INSERT INTO
                                    emp (empno, ename)
                                            VALUES
                                    (:empno,:ename)
                            RETURNING
                                    ROWID
                                 INTO
                                    :rid
                                        ");

$data = array(
              1111 => "Larry",
              2222 => "Bill",
              3333 => "Jim"
             );

$rowid = oci_new_descriptor($conn, OCI_D_ROWID);

oci_bind_by_name($stmt, ":empno", $empno, 32);
oci_bind_by_name($stmt, ":ename", $ename, 32);
oci_bind_by_name($stmt, ":rid",   $rowid, -1, OCI_B_ROWID);

$update = oci_parse($conn, "
                             UPDATE
                                   emp
                               SET
                                   sal = :sal
                              WHERE
                                   ROWID = :rid
                             ");
oci_bind_by_name($update, ":rid", $rowid, -1, OCI_B_ROWID);
oci_bind_by_name($update, ":sal", $sal,   32);

$sal = 10000;

while (list($empno, $ename) = each($data)) {
    oci_execute($stmt);
 oci_execute($update);
}

$rowid->free();

oci_free_statement($update);
oci_free_statement($stmt);

$stmt = oci_parse($conn, "
                          SELECT
                                *
                            FROM
                                emp
                           WHERE
                                empno
                              IN
                                (1111,2222,3333)
                              ");
oci_execute($stmt);

while ($row = oci_fetch_assoc($stmt)) {
    var_dump($row);
}

oci_free_statement($stmt);
```

Remember, that this function strips trailing whitespace. See the following example:

Ejemplo 2. oci_bind_by_name() example

```
<?php
    $connection = oci_connect('apelsin','kanistra');
    $query = "INSERT INTO test_table VALUES(:id, :text)";

    $statement = oci_parse($query);
    oci_bind_by_name($statement, ":id", 1);
    oci_bind_by_name($statement, ":text", "trailing spaces follow     ");
    oci_execute($statement);
    /*
     This code will insert into DB string 'trailing spaces follow', without
     trailing spaces
    */
?>
```

Ejemplo 3. oci_bind_by_name() example

```
<?php
    $connection = oci_connect('apelsin','kanistra');
    $query = "INSERT INTO test_table VALUES(:id, 'trailing spaces follow      ')";

    $statement = oci_parse($query);
    oci_bind_by_name($statement, ":id", 1);
    oci_execute($statement);
    /*
     And this code will add 'trailing spaces follow      ', preserving
     trailing whitespaces
    */
?>
```

Aviso
Do not use magic_quotes_gpc or **addslashes()** and **oci_bind_by_name()** simultaneously as no quoting is needed and any magically applied quotes will be written into your database as **oci_bind_by_name()** is not able to distinguish magically added quotings from those added intentionally.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

> **Nota:** In PHP versions before 5.0.0 you must use **ocibindbyname()** instead. This name still can be used, it was left as alias of **oci_bind_by_name()** for downwards compatability. This, however, is deprecated and not recommended.

oci_cancel

(PHP 5)

oci_cancel -- Cancels reading from cursor

Description

bool **oci_cancel** (resource stmt)

oci_cancel() invalidates a cursor, freeing all associated resources and cancels the ability to read from it.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

In PHP versions before 5.0.0 you must use **ocicancel()** instead. This name still can be used, it was

left as alias of **oci_cancel()** for downwards compatability. This, however, is deprecated and not recommended.

oci_close

(PHP 5)

oci_close -- Closes Oracle connection

Description

bool **oci_close** (resource connection)

oci_close() closes the Oracle connection *connection*.

> **Nota:** As non-persistent links are closed automatically at the end of script execution, calling this function is not required. Because of this and the method the extension uses to handle connection resources, **oci_close()** currently provides no actual functionality.

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

> **Nota:** In PHP versions before 5.0.0 you must use **ociclose()** instead. This name still can be used, it was left as alias of **oci_close()** for downwards compatability. This, however, is deprecated and not recommended.

OCI-Collection->append

(no version information, might be only in CVS)

OCI-Collection->append -- Appends an object to the collection

Description

bool **OCI-Collection->append** (mixed value)

Appends an object to the end of the collection. Parameter *value* can be a string or a number.

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

OCI-Collection->assign

(no version information, might be only in CVS)

OCI-Collection->assign -- Assigns a value to the collection from another existing collection

Description

bool **OCI-Collection->assign** (OCI-Collection from)

Assigns a value to the collection from another, previously created collection. Both collections must be created with **oci_new_collection()** prior to using them.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

OCI-Collection->assignElem

(no version information, might be only in CVS)

OCI-Collection->assignElem -- Assigns a value to the element of the collection

Description

bool **OCI-Collection->assignElem** (int index, mixed value)

Assigns a value to the element with index *index*. Parameter *value* can be a string or a number.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

OCI-Collection->getElem

(no version information, might be only in CVS)

OCI-Collection->getElem -- Returns value of the element

Description

mixed **OCI-Collection->getElem** (int index)

Method **OCI-Collection->getElem()** returns value of the element with index *index* (1-based).

OCI-Collection->getElem() will return **FALSE** if such element doesn't exist; **NULL** if element is **NULL**; string if element is column of a string datatype or number if element is numeric field.

OCI-Collection->getElem() will return **FALSE** in case of error.

OCI-Collection->free

(no version information, might be only in CVS)

OCI-Collection->free -- Frees resources associated with collection object

Description

bool **OCI-Collection->free** (void)

Frees resources associated with collection object.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

OCI-Collection->max

(no version information, might be only in CVS)

OCI-Collection->max -- Gets the maximum number of elements in the collection

Description

int **OCI-Collection->max** (void)

Returns the maximum number of elements in the collection. If the returned value is 0, then the number of elements is not limited. **OCI-Collection->max()** returns **`FALSE`** in case of error.

See also **oci_collection_size()**.

OCI-Collection->size

(no version information, might be only in CVS)

OCI-Collection->size -- Returns size of the collection

Description

int **OCI-Collection->size** (void)

Returns the number of elements in the collection.

See also **oci_collection_max()**.

OCI-Collection->trim

(no version information, might be only in CVS)

OCI-Collection->trim -- Trims elements from the end of the collection

Description

bool **OCI-Collection->trim** (int num)

Trims *num* of elements from the end of the collection.

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

See also **oci_collection_size()**.

oci_commit

(PHP 5)

oci_commit -- Commits outstanding statements

Description

bool **oci_commit** (resource connection)

oci_commit() commits all outstanding statements for the active transaction on the Oracle connection *connection*.

Ejemplo 1. oci_commit() example

```
<?php
    // Login to Oracle server
    $conn = oci_connect('scott', 'tiger');

    // Parse SQL
    $stmt = oci_parse($conn, "
                              INSERT INTO
                                        employees (name, surname)
                                   VALUES
                                        ('Maxim', 'Maletsky')
                             ");

    /* Execute statement
       OCI_DEFAULT tells oci_execute()
       not to commit statement immediately */
    oci_execute($stmt, OCI_DEFAULT);

    /*
    ....
    Parsing and executing other statements here ...
    ....
    */

    // Commit transaction
    $committed = oci_commit($conn);

    // Test whether commit was successful. If error occurred, return error message
    if (!$committed) {
        $error = oci_error($conn);
        echo 'Commit failed. Oracle reports: ' . $error['message'];
    }

?>
```

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

> **Nota:** In PHP versions before 5.0.0 you must use **ocicommit()** instead. This name still can be used, it was left as alias of **oci_commit()** for downwards compatability. This, however, is deprecated and not recommended.

See also **oci_rollback()** and **oci_execute()**.

oci_connect

(PHP 5)

oci_connect -- Establishes a connection to Oracle server

Description

resource **oci_connect** (string username, string password [, string db [, string charset]])

oci_connect() returns a connection identifier needed for most other OCI calls. The optional third parameter can either contain the name of the local Oracle instance or the name of the entry in `tnsnames.ora` to which you want to connect. If the optional third parameter is not specified, PHP uses the environment variables **`ORACLE_SID`** (Oracle instance) or **`TWO_TASK`** (`tnsnames.ora`) to determine which database to connect to.

> **Nota: oci_connect()** *does not* reestablish the connection, if a connection with such parameters was established before. In this case, **oci_connect()** will return identifier of previously opened connection. This means, that you cannot use this function to separate transactions. To establish a distinctly new connection, use **oci_new_connect()**.

Si usais Oracle server v.9.2 ó superior, podeis definir el parametro *charset*, el cual será utilizado en la nueva conexión. Si utilizais una version < 9.2, este parámetro será ignorado y la variable de entorno NLS_LANG será usada en su lugar.

Ejemplo 1. oci_connect() example

```php
<?php
echo "<pre>";
$db = "";

$c1 = oci_connect("scott", "tiger", $db);
$c2 = oci_connect("scott", "tiger", $db);

function create_table($conn)
{
  $stmt = oci_parse($conn, "create table scott.hallo (test varchar2(64))");
  oci_execute($stmt);
  echo $conn . " created table\n\n";
}

function drop_table($conn)
{
  $stmt = oci_parse($conn, "drop table scott.hallo");
  oci_execute($stmt);
  echo $conn . " dropped table\n\n";
}

function insert_data($conn)
{
  $stmt = oci_parse($conn, "insert into scott.hallo
            values('$conn' || ' ' || to_char(sysdate,'DD-MON-YY HH24:MI:SS'))");
  oci_execute($stmt, OCI_DEFAULT);
  echo $conn . " inserted hallo\n\n";
}

function delete_data($conn)
{
  $stmt = oci_parse($conn, "delete from scott.hallo");
  oci_execute($stmt, OCI_DEFAULT);
  echo $conn . " deleted hallo\n\n";
}

function commit($conn)
{
  oci_commit($conn);
  echo $conn . " committed\n\n";
}

function rollback($conn)
{
  oci_rollback($conn);
  echo $conn . " rollback\n\n";
}

function select_data($conn)
{
  $stmt = oci_parse($conn, "select * from scott.hallo");
  oci_execute($stmt, OCI_DEFAULT);
  echo $conn."----selecting\n\n";
  while (oci_fetch($stmt)) {
    echo $conn . " [" . oci_result($stmt, "TEST") . "]\n\n";
  }
  echo $conn . "----done\n\n";
}

create_table($c1);
insert_data($c1);   // Insert a row using c1
insert_data($c2);   // Insert a row using c2

select_data($c1);   // Results of both inserts are returned
select_data($c2);

rollback($c1);      // Rollback using c1

select_data($c1);   // Both inserts have been rolled back
select_data($c2);

insert_data($c2);   // Insert a row using c2
```

oci_connect() returns **FALSE** if an error occured.

> **Nota:** In PHP versions before 5.0.0 you must use **ocilogon()** instead. This name still can be used, it was left as alias of **oci_connect()** for downwards compatability. This, however, is deprecated and not recommended.

See also **oci_pconnect()** and **oci_new_connect()**.

oci_define_by_name

(PHP 5)

oci_define_by_name -- Uses a PHP variable for the define-step during a SELECT

Description

bool **oci_define_by_name** (resource statement, string column_name, mixed &variable [, int type])

oci_define_by_name() defines PHP variables for fetches of SQL-Columns. Be careful that Oracle uses ALL-UPPERCASE column names, whereby in your select you can also write lowercase. **oci_define_by_name()** expects the *column_name* to be in uppercase. If you define a variable that doesn't exists in your select statement, no error will issued.

If you need to define an abstract datatype (LOB/ROWID/BFILE) you must allocate it first using **oci_new_descriptor()**. See also the **oci_bind_by_name()** function.

Ejemplo 1. oci_define_by_name() example

```
<?php
/* oci_define_by_name example - thies at thieso dot net (980219) */

$conn = oci_connect("scott", "tiger");

$stmt = oci_parse($conn, "SELECT empno, ename FROM emp");

/* the define MUST be done BEFORE oci_execute! */

oci_define_by_name($stmt, "EMPNO", $empno);
oci_define_by_name($stmt, "ENAME", $ename);

oci_execute($stmt);

while (oci_fetch($stmt)) {
    echo "empno:" . $empno . "\n";
    echo "ename:" . $ename . "\n";
}

oci_free_statement($stmt);
oci_close($conn);
?>
```

> **Nota:** In PHP versions before 5.0.0 you must use **ocidefinebyname()** instead. This name still can be used, it was left as alias of **oci_define_by_name()** for downwards compatability. This, however, is deprecated and not recommended.

oci_error

(PHP 5)

oci_error -- Returns the last error found

Description

array **oci_error** ([resource source])

For most errors, the parameter is the most appropriate resource handle. For connection errors with **oci_connect()**, **oci_new_connect()** or **oci_pconnect()**, do not pass a parameter. If no error is found, **oci_error()** returns **FALSE**. **oci_error()** returns the error as an associative array. In this array, *code* consists the oracle error code and *message* the oracle error string.

> **As of PHP 4.3:** *offset* and *sqltext* will also be included in the return array to indicate the location of the error and the original SQL text which caused it.

Ejemplo 1. Displaying the Oracle error message after a connection error

```
$conn = @oci_connect("scott", "tiger", "mydb");
if (!$conn) {
  $e = oci_error();   // For oci_connect errors pass no handle
  echo htmlentities($e['message']);
}
```

Ejemplo 2. Displaying the Oracle error message after a parsing error

```
$stmt = @oci_parse($conn, "select ' from dual");  // note mismatched quote
if (!$stmt) {
  $e = oci_error($conn);  // For oci_parse errors pass the connection handle
  echo htmlentities($e['message']);
}
```

Ejemplo 3. Displaying the Oracle error message and problematic statement after an execution error

```
$r = oci_execute($stmt);
if (!$r) {
  $e = oci_error($stmt); // For oci_execute errors pass the statementhandle
  echo htmlentities($e['message']);
  echo "<pre>";
  echo htmlentities($e['sqltext']);
  printf("\n%".($e['offset']+1)."s", "^");
  echo "</pre>";
}
```

> **Nota:** In PHP versions before 5.0.0 you must use **ocierror()** instead. This name still can be used, it was left as alias of **oci_error()** for downwards compatability. This, however, is deprecated and not recommended.

oci_execute

(PHP 5)

oci_execute -- Executes a statement

Description

bool **oci_execute** (resource stmt [, int mode])

oci_execute() executes a previously parsed statement (see **oci_parse()**). The optional *mode* allows you to specify the execution mode (default is **OCI_COMMIT_ON_SUCCESS**). If you don't want statements to be committed automatically, you should specify **OCI_DEFAULT** as your *mode*.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

> **Nota:** In PHP versions before 5.0.0 you must use **ociexecute()** instead. This name still can be used, it was left as alias of **oci_execute()** for downwards compatability. This, however, is deprecated and not recommended.

oci_fetch_all

(PHP 5)

oci_fetch_all -- Fetches all rows of result data into an array

Description

int **oci_fetch_all** (resource statement, array &output [, int skip [, int maxrows [, int flags]]])

oci_fetch_all() fetches all the rows from a result into a user-defined array. **oci_fetch_all()** returns the number of rows fetched or **FALSE** in case of error. *skip* is the number of initial rows to ignore when fetching the result (default value of 0, to start at the first line). *maxrows* is the number of rows to read, starting at the *skip*th row (default to -1, meaning all the rows).

> **Nota:** Esta funcion define campos NULL como valores PHP **NULL**.

Parameter *flags* can be any combination of the following:

OCI_FETCHSTATEMENT_BY_ROW
OCI_FETCHSTATEMENT_BY_COLUMN (default value)
OCI_NUM
OCI_ASSOC
Ejemplo 1. oci_fetch_all() example

```
<?php
/* oci_fetch_all example mbritton at verinet dot com (990624) */

$conn = oci_connect("scott", "tiger");

$stmt = oci_parse($conn, "select * from emp");

oci_execute($stmt);

$nrows = oci_fetch_all($stmt, $results);
if ($nrows > 0) {
   echo "<table border=\"1\">\n";
   echo "<tr>\n";
   while (list($key, $val) = each($results)) {
      echo "<th>$key</th>\n";
   }
   echo "</tr>\n";

   for ($i = 0; $i < $nrows; $i++) {
      reset($results);
      echo "<tr>\n";
      while ($column = each($results)) {
         $data = $column['value'];
         echo "<td>$data[$i]</td>\n";
      }
      echo "</tr>\n";
   }
   echo "</table>\n";
} else {
   echo "No data found<br />\n";
}
echo "$nrows Records Selected<br />\n";

oci_free_statement($stmt);
oci_close($conn);
?>
```

oci_fetch_all() returns **FALSE** in case of error.

> **Nota:** In PHP versions before 5.0.0 you must use **ocifetchstatement()** instead. This name still can be used, it was left as alias of **oci_fetch_all()** for downwards compatability. This, however, is deprecated and not recommended.

oci_fetch_array

(PHP 5)

oci_fetch_array -- Returns the next row from the result data as an associative or numeric array, or both

Description

array **oci_fetch_array** (resource statement [, int mode])

Returns an array, which corresponds to the next result row or **FALSE** in case of error or there is no more rows in the result.

oci_fetch_array() returns an array with both associative and numeric indices.

> **Nota:** Esta funcion define campos NULL como valores PHP **NULL**.

Optional second parameter can be any combination of the following constants:

OCI_BOTH - return an array with both associative and numeric indices (the same as **OCI_ASSOC** + **OCI_NUM**). This is the default behavior.
OCI_ASSOC - return an associative array (as **oci_fetch_assoc()** works).
OCI_NUM - return a numeric array, (as **oci_fetch_row()** works).
OCI_RETURN_NULLS - create empty elements for the **NULL** fields.
OCI_RETURN_LOBS - return the value of a LOB of the descriptor.
Default *mode* is **OCI_BOTH**.

It should be mentioned here, that **oci_fetch_array()** is *insignificantly* slower, than **oci_fetch_row()**, but much more handy.

> **Nota:** Don't forget, that Oracle returns all field names in uppercase and associative indices in the result array will be uppercased too.

Ejemplo 1. oci_fetch_array() with OCI_BOTH example

```
<?php
$connection = oci_connect("apelsin", "kanistra");

$query = "SELECT id, name FROM fruits";

$statement = oci_parse ($connection, $query);
oci_execute ($statement);

while ($row = oci_fetch_array ($statement, OCI_BOTH)) {
    echo $row[0]." and ".$row['ID']." is the same<br>";
    echo $row[1]." and ".$row['NAME']." is the same<br>";
}
?>
```

Ejemplo 2. oci_fetch_array() with OCI_NUM example

```
<?php
$connection = oci_connect("user", "password");

$query = "SELECT id, name, lob_field FROM fruits";

$statement = oci_parse ($connection, $query);
oci_execute ($statement);

while ($row = oci_fetch_array ($statement, OCI_NUM)) {
    echo $row[0]."<br>";
    echo $row[1]."<br>";
    echo $row[2]->read(100)."<br>";  //this will output first 100 bytes from LOB
}
?>
```

Ejemplo 3. oci_fetch_array() with OCI_ASSOC example

```
<?php
$connection = oci_connect("user", "password");

$query = "SELECT id, name, lob_field FROM fruits";

$statement = oci_parse ($connection, $query);
oci_execute ($statement);

while ($row = oci_fetch_array ($statement, OCI_NUM)) {
    echo $row['ID']."<br>";
    echo $row['NAME']."<br>";
    echo $row['LOB_FIELD']."<br>";  //this will output "Object id #1"
}
?>
```

Ejemplo 4. oci_fetch_array() with `OCI_RETURN_LOBS` example

```
<?php
$connection = oci_connect("user", "password");

$query = "SELECT id, name, lob_field FROM fruits";

$statement = oci_parse ($connection, $query);
oci_execute ($statement);

while ($row = oci_fetch_array ($statement, OCI_NUM)) {
    echo $row[0]."<br>";
    echo $row[1]."<br>";
    echo $row['LOB_FIELD']."<br>";  //this will output LOB's content
}
?>
```

See also **oci_fetch_assoc()**, **oci_fetch_object()**, **oci_fetch_row()** and **oci_fetch_all()**.

oci_fetch_assoc

(PHP 5)

oci_fetch_assoc -- Returns the next row from the result data as an associative array

Description

array **oci_fetch_assoc** (resource statement)

oci_fetch_assoc() returns the next row from the result data as an associative array (identical to **oci_fetch_array()** call with **`OCI_ASSOC`** flag).

> **Nota:** Esta funcion define campos NULL como valores PHP **`NULL`**.

Subsequent call to **oci_fetch_assoc()** will return next row or **`FALSE`** if there is no more rows.

> **Nota:** Don't forget, that Oracle returns all field names in uppercase and associative indices in the result array will be uppercased too.

See also **oci_fetch_array()**, **oci_fetch_object()**, **oci_fetch_row()** and **oci_fetch_all()**.

oci_fetch_object

(PHP 5)

oci_fetch_object -- Returns the next row from the result data as an object

Description

object **oci_fetch_object** (resource statement)

oci_fetch_object() returns the next row from the result data as an object, which attributes correspond to fields in statement.

> **Nota:** Esta funcion define campos NULL como valores PHP **`NULL`**.

Subsequent calls to **oci_fetch_object()** will return the next row from the result or **FALSE** if there is no more rows.

> **Nota:** Don't forget, that Oracle returns all field names in uppercase and attributes' names in the result object will be in uppercase as well.

See also **oci_fetch_array()**, **oci_fetch_assoc()**, **oci_fetch_row()** and **oci_fetch_all()**.

oci_fetch_row

(PHP 5)

oci_fetch_row -- Returns the next row from the result data as a numeric array

Description

array **oci_fetch_row** (resource statement)

Calling **oci_fetch_row()** is identical to **oci_fetch_array()** with **OCI_NUM** flag and returns the next row from the result data as a numeric array.

> **Nota:** Esta funcion define campos NULL como valores PHP **NULL**.

Subsequent calls to **oci_fetch_row()** will return the next row from the result data or **FALSE** if there is no more rows.

See also **oci_fetch_array()**, **oci_fetch_object()**, **oci_fetch_assoc()** and **oci_fetch_all()**.

oci_fetch

(PHP 5)

oci_fetch -- Fetches the next row into result-buffer

Description

bool **oci_fetch** (resource statement)

oci_fetch() fetches the next row (for SELECT statements) into the internal result-buffer.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

> **Nota:** In PHP versions before 5.0.0 you must use **ocifetch()** instead. This name still can be used, it was left as alias of **oci_fetch()** for downwards compatability. This, however, is deprecated and not recommended.

oci_field_is_null

(PHP 5)

oci_field_is_null -- Checks if the field is **NULL**

Description

bool **oci_field_is_null** (resource stmt, mixed field)

oci_field_is_null() returns **TRUE** if field *field* from the *statement* is **NULL**. Parameter *field* could be a field's index or a field's name (uppercased).

> **Nota:** In PHP versions before 5.0.0 you must use **ocicolumnisnull()** instead. This name still can be used, it was left as alias of **oci_field_is_null()** for downwards compatability. This, however, is deprecated and not recommended.

oci_field_name

(PHP 5)

oci_field_name -- Returns the name of a field from the statement

Description

string **oci_field_name** (resource statement, int field)

oci_field_name() returns the name of the field corresponding to the field number (1-based).

Ejemplo 1. oci_field_name() example

```
<?php
    $conn = oci_connect("scott", "tiger");
    $stmt = oci_parse($conn, "SELECT * FROM emp");
    oci_execute($stmt);

    echo "<table border=\"1\">";
    echo "<tr>";
    echo "<th>Name</th>";
    echo "<th>Type</th>";
    echo "<th>Length</th>";
    echo "</tr>";

    $ncols = oci_num_fields($stmt);

    for ($i = 1; $i <= $ncols; $i++) {
        $column_name  = oci_field_name($stmt, $i);
        $column_type  = oci_field_type($stmt, $i);
        $column_size  = oci_field_size($stmt, $i);

        echo "<tr>";
        echo "<td>$column_name</td>";
        echo "<td>$column_type</td>";
        echo "<td>$column_size</td>";
        echo "</tr>";
    }

    echo "</table>\n";
    oci_free_statement($stmt);
    oci_close($conn);
?>
```

> **Nota:** In PHP versions before 5.0.0 you must use **ocicolumnname()** instead. This name still can be used, it was left as alias of **oci_field_name()** for downwards compatability.

This, however, is deprecated and not recommended.

See also **oci_num_fields()**, **oci_field_type()**, and **oci_field_size()**.

oci_field_precision

(PHP 5)

oci_field_precision -- Tell the precision of a field

Description

int **oci_field_precision** (resource statement, int field)

Returns precision of the field with *field* index (1-based).

For FLOAT columns, precision is nonzero and scale is -127. If precision is 0, then column is NUMBER. Else it's NUMBER(precision, scale).

> **Nota:** In PHP versions before 5.0.0 you must use **ocicolumnprecision()** instead. This name still can be used, it was left as alias of **oci_field_precision()** for downwards compatability. This, however, is deprecated and not recommended.

See also **oci_field_scale()** and **oci_field_type()**.

oci_field_scale

(PHP 5)

oci_field_scale -- Tell the scale of the field

Description

int **oci_field_scale** (resource statement, int field)

Returns scale of the column with *field* index (1-based) or **`FALSE`** if there is no such field.

For FLOAT columns, precision is nonzero and scale is -127. If precision is 0, then column is NUMBER. Else it's NUMBER(precision, scale).

> **Nota:** In PHP versions before 5.0.0 you must use **ocicolumnscale()** instead. This name still can be used, it was left as alias of **oci_field_scale()** for downwards compatability. This, however, is deprecated and not recommended.

See also **oci_field_precision()** and **oci_field_type()**.

oci_field_size

(PHP 5)

oci_field_size -- Returns field's size

Description

int **oci_field_size** (resource stmt, mixed field)

oci_field_size() returns the size of a field in bytes. Value of *field* parameter can be the field's index (1-based) or it's name.

Ejemplo 1. oci_field_size()example

```
<?php
    $conn = oci_connect("scott", "tiger");
    $stmt = oci_parse($conn, "SELECT * FROM emp");
    oci_execute($stmt);

    echo "<table border=\"1\">";
    echo "<tr>";
    echo "<th>Name</th>";
    echo "<th>Type</th>";
    echo "<th>Length</th>";
    echo "</tr>";

    $ncols = oci_num_fields($stmt);

    for ($i = 1; $i <= $ncols; $i++) {
        $column_name  = oci_field_name($stmt, $i);
        $column_type  = oci_field_type($stmt, $i);
        $column_size  = oci_field_size($stmt, $i);
        echo "<tr>";
        echo "<td>$column_name</td>";
        echo "<td>$column_type</td>";
        echo "<td>$column_size</td>";
        echo "</tr>";
    }

    echo "</table>";

    oci_free_statement($stmt);
    oci_close($conn);
?>
```

> **Nota:** In PHP versions before 5.0.0 you must use **ocicolumnsize()** instead. This name still can be used, it was left as alias of **oci_field_size()** for downwards compatability. This, however, is deprecated and not recommended.

See also **oci_num_fields()** and **oci_field_name()**.

oci_field_type_raw

(PHP 5)

oci_field_type_raw -- Tell the raw Oracle data type of the field

Description

int **oci_field_type_raw** (resource statement, int field)

oci_field_type_raw() returns Oracle's raw data type of the field.

> **Nota:** In PHP versions before 5.0.0 you must use **ocicolumntyperaw()** instead. This name still can be used, it was left as alias of **oci_field_type_raw()** for downwards compatability. This, however, is deprecated and not recommended.

However, if you want to get field's type, then **oci_field_type()** will suit you better. See **oci_field_type()** for additional information.

oci_field_type

(PHP 5)

oci_field_type -- Returns field's data type

Description

mixed **oci_field_type** (resource stmt, int field)

oci_field_type() returns a field's data type. Parameter *field* is an index of the field in the statement (1-based).

Ejemplo 1. oci_field_type() example

```
<?php
    $conn = oci_connect("scott", "tiger");
    $stmt = oci_parse($conn, "SELECT * FROM emp");
    oci_execute($stmt);

    echo "<table border=\"1\">";
    echo "<tr>";
    echo "<th>Name</th>";
    echo "<th>Type</th>";
    echo "<th>Length</th>";
    echo "</tr>";

    $ncols = oci_num_fields($stmt);

    for ($i = 1; $i <= $ncols; $i++) {
        $column_name  = oci_field_name($stmt, $i);
        $column_type  = oci_field_type($stmt, $i);
        $column_size  = oci_field_size($stmt, $i);

        echo "<tr>";
        echo "<td>$column_name</td>";
        echo "<td>$column_type</td>";
        echo "<td>$column_size</td>";
        echo "</tr>";
    }

    echo "</table>\n";

    oci_free_statement($stmt);
    oci_close($conn);
?>
```

> **Nota:** In PHP versions before 5.0.0 you must use **ocicolumntype()** instead. This name still can be used, it was left as alias of **oci_field_type()** for downwards compatability. This, however, is deprecated and not recommended.

See also **oci_num_fields()**, **oci_field_name()**, and **oci_field_size()**.

descriptor->free

(no version information, might be only in CVS)

descriptor->free -- Frees resources associated with descriptor

Description

bool **descriptor->free** (void)

descriptor->free() frees resources associated with descriptor, previously allocated with **oci_new_descriptor()**.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

oci_free_statement

(PHP 5)

oci_free_statement -- Frees all resources associated with statement or cursor

Description

bool **oci_free_statement** (resource statement)

oci_free_statement() frees resources associated with Oracle's cursor or statement, which was received from as a result of **oci_parse()** or obtained from Oracle.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

oci_internal_debug

(PHP 5)

oci_internal_debug -- Enables or disables internal debug output

Description

void **oci_internal_debug** (int onoff)

oci_internal_debug() enables or disables internal debug output. Set *onoff* to 0 to turn debug output off or 1 to turn it on.

> **Nota:** In PHP versions before 5.0.0 you must use **ociinternaldebug()** instead. This name still can be used, it was left as alias of **oci_internal_debug()** for downwards compatability. This, however, is deprecated and not recommended.

lob->append

(no version information, might be only in CVS)

lob->append -- Appends data from the large object to another large object

Description

bool **lob->append** (OCI-Lob lob_from)

Appends data from the large object to the end of another large object.

Writing to the large object with **lob->append()** will fail if buffering was previously enabled. You must disable buffering before appending. You may need to flush buffers with **oci_lob_flush()** before disabling buffering.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

See also **oci_lob_flush()**, **ocisetbufferinglob()** and **ocigetbufferinglob()**.

lob->close

(no version information, might be only in CVS)

lob->close -- Closes LOB descriptor

Description

bool **lob->close** (void)

lob->close() closes descriptor of LOB or FILE. This function should be used only with **lob->writeTemporary()**.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

> **Nota:** In PHP versions before 5.0.0 you must use **ocicloselob()** instead. This name still can be used, it was left as alias of **oci_lob_close()** for downwards compatability. This, however, is deprecated and not recommended.

See also **oci_lob_write_temporary()**.

oci_lob_copy

(PHP 5)

oci_lob_copy -- Copies large object

Description

bool **oci_lob_copy** (OCI-Lob lob_to, OCI-Lob lob_from [, int length])

Copies large object or a part of large object to another large object. Parameter *length* indicates the length of data to be copied. Old data of LOB-recipient will be overwritten.

If you need to copy a particular part of LOB to a particular position of LOB, you can use **oci_lob_seek()** to move internal pointers of LOBs.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

lob->eof

(no version information, might be only in CVS)

lob->eof -- Tests for end-of-file on a large object's descriptor

Description

bool **lob->eof** (void)

Returns **TRUE** if internal pointer of large object is at the end of LOB. Otherwise returns **FALSE**.

See also **oci_lob_size()**.

lob->erase

(no version information, might be only in CVS)

lob->erase -- Erases a specified portion of the internal LOB data

Description

int **lob->erase** ([int offset [, int length]])

Erases a specified portion of the internal LOB data starting at a specified *offset*. Parameters *length* and *offset* are optional. **lob->erase()** erases all LOB data by default.

For BLOBs, erasing means that the existing LOB value is overwritten with zero-bytes. For CLOBs, the existing LOB value is overwritten with spaces.

lob->erase() returns the actual number of characters/bytes erased or **FALSE** in case of error.

lob->export

(no version information, might be only in CVS)

lob->export -- Exports LOB's contents to a file

Description

bool **lob->export** (string filename [, int start [, int length]])

Exports LOB's contents to a file, which name is given in parameter *filename*. Optional parameter *start* indicates from what position to start export and parameter *length* - length of data to be exported.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

lob->flush

(no version information, might be only in CVS)

lob->flush -- Flushes/writes buffer of the LOB to the server

Description

bool **lob->flush** ([int flag])

lob->flush() actually writes data to the server. By default, resources are not freed, but using flag **OCI_LOB_BUFFER_FREE** you can do it explicitly. Be sure you know what you're doing - next read/write operation to the same part of LOB will involve a round-trip to the server and initialize new buffer resources. Tt is recommended to use **OCI_LOB_BUFFER_FREE** flag only when you are not going to work with the LOB anymore.

lob->flush() returns **FALSE** if buffering was not enabled or an error occurred.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

lob->import

(no version information, might be only in CVS)

lob->import -- Imports file data to the LOB

Description

bool **lob->import** (string filename)

Writes data from *filename* in to the current position of large object.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

> **Nota:** In PHP versions before 5.0.0 you must use **ocisavelobfile()** instead. This name still can be used, it was left as alias of **oci_lob_import()** for downwards compatability. This, however, is deprecated and not recommended.

oci_lob_is_equal

(PHP 5)

oci_lob_is_equal -- Compares two LOB/FILE locators for equality

Description

bool **oci_lob_is_equal** (OCI-Lob lob1, OCI-Lob lob2)

Compares two LOB/FILE locators. Returns **TRUE** if these objects are equal and **FALSE** otherwise.

lob->load

(no version information, might be only in CVS)

lob->load -- Returns large object's contents

Description

string **lob->load** (void)

Returns large object's contents. As script execution is terminated when the memory_limit is reached, ensure that the LOB does not exceed this limit. In most cases it's recommended to use **oci_lob_read()** instead. In case of error **lob->load()** returns **FALSE**.

> **Nota:** In PHP versions before 5.0.0 you must use **ociloadlob()** instead. This name still can be used, it was left as alias of **oci_lob_load()** for downwards compatability. This, however, is deprecated and not recommended.

lob->read

(no version information, might be only in CVS)

lob->read -- Reads part of large object

Description

string **lob->read** (int length)

Reads *length* bytes from the current position of LOB's internal pointer. Reading stops when *length* bytes have been read or end of large object is reached. Internal pointer of large object will be shifted on the amount of bytes read.

Returns **FALSE** in case of error.

See also **oci_lob_eof()**, **oci_lob_seek()**, **oci_lob_tell()** and **oci_lob_write()**.

lob->rewind

(no version information, might be only in CVS)

lob->rewind -- Moves the internal pointer to the beginning of the large object

Description

bool **lob->rewind** (void)

Sets the internal pointer to the beginning of the large object.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

See also **oci_lob_seek()** and **oci_lob_tell()**.

lob->save

(no version information, might be only in CVS)

lob->save -- Saves data to the large object

Description

bool **lob->save** (string data [, int offset])

Saves *data* to the large object. Parameter *offset* can be used to indicate offset from the beginning of the large object.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

> **Nota:** In PHP versions before 5.0.0 you must use **ocisavelob()** instead. This name still can be used, it was left as alias of **oci_lob_save()** for downwards compatability. This, however, is deprecated and not recommended.

See also **oci_lob_write()** and **oci_lob_import()**.

lob->seek

(no version information, might be only in CVS)

lob->seek -- Sets the internal pointer of the large object

Description

bool **lob->seek** (int offset [, int whence])

Sets the internal pointer of the large object. Parameter *offset* indicates the amount of bytes, on which internal pointer should be moved from the position, pointed by *whence*:

OCI_SEEK_SET - sets the position equal to *offset*
OCI_SEEK_CUR - adds *offset* bytes to the current position
OCI_SEEK_END - adds *offset* bytes to the end of large object (use negative value to move to a position before the end of large object)
See also **oci_lob_rewind()** and **oci_lob_tell()**.

lob->size

(no version information, might be only in CVS)

lob->size -- Returns size of large object

Description

int **lob->size** (void)

Returns length of large object value or **FALSE** in case of error. Empty objects have zero length.

lob->tell

(no version information, might be only in CVS)

lob->tell -- Returns current position of internal pointer of large object

Description

int **lob->tell** (void)

Returns current position of a LOB's internal pointer or **FALSE** if an error occurred.

See also **oci_lob_size()** and **oci_lob_eof()**.

lob->truncate

(no version information, might be only in CVS)

lob->truncate -- Truncates large object

Description

bool **lob->truncate** ([int length])

If parameter *length* is given, **lob->truncate()** truncates large object to *length* bytes. Otherwise, **lob->truncate()** will purge the LOB completely.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

See also **oci_lob_erase()**.

lob->writeTemporary

(no version information, might be only in CVS)

lob->writeTemporary -- Writes temporary large object

Description

bool **lob->writeTemporary** (string data [, int lob_type])

Creates a temporary large object and writes *data* to it.

Parameter *lob_type* can be one of the following:

OCI_TEMP_BLOB is used to create temporary BLOBs
OCI_TEMP_CLOB is used to create temporary CLOBs
lob->writeTemporary() creates a CLOB by default.

You should use **oci_lob_close()** when the work with the object is over.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

> **Nota:** In PHP versions before 5.0.0 you must use **ociwritetemporarylob()** instead. This name still can be used, it was left as alias of **oci_lob_write_temporary()** for downwards compatability. This, however, is deprecated and not recommended.

See also **oci_lob_close()**.

lob->write

(no version information, might be only in CVS)

lob->write -- Writes data to the large object

Description

int **lob->write** (string data [, int length])

Writes data from the parameter *data* into the current position of LOB's internal pointer. If the parameter *length* is given, writing will stop after *length* bytes have been written or the end of *data* is reached, whichever comes first.

lob->write() returns the number of bytes written or **FALSE** in case of error.

See also **oci_lob_read()**.

oci_new_collection

(PHP 5)

oci_new_collection -- Allocates new collection object

Description

OCI-Collection **oci_new_collection** (resource connection, string tdo [, string schema])

Allocates new collection object. Parameter *tdo* should be a valid named type (uppercased). Third, optional parameter *schema* should point to the scheme, where the named type was created. **oci_new_collection()** uses name of the current user as default value of *schema*.

oci_new_collection() returns **`FALSE`** on error.

> **Nota:** In PHP versions before 5.0.0 you must use **ocinewcollection()** instead. This name still can be used, it was left as alias of **oci_new_collection()** for downwards compatability. This, however, is deprecated and not recommended.

oci_new_connect

(PHP 5)

oci_new_connect -- Establishes a new connection to the Oracle server

Description

resource **oci_new_connect** (string username, string password [, string db [, string charset]])

oci_new_connect() creates a new connection to an Oracle server and logs on. The optional third parameter can either contain the name of the local Oracle instance or the name of the entry in `tnsnames.ora`. If the third parameter is not specified, PHP uses environment variables *ORACLE_SID* and *TWO_TASK* to determine the name of local Oracle instance and location of `tnsnames.ora` accordingly.

Si usais Oracle server v.9.2 ó superior, podeis definir el parametro *charset*, el cual será utilizado en la nueva conexión. Si utilizais una version < 9.2, este parámetro será ignorado y la variable de entorno NLS_LANG será usada en su lugar.

oci_new_connect() forces the creation of a new connection. This should be used if you need to isolate a set of transactions. By default, connections are shared and subsequent calls to **oci_connect()** will return the same connection identifier.

The following demonstrates how you can separate connections.
Ejemplo 1. oci_new_connect() example

```
<?php
echo "<html><pre>";
$db = "";

$c1 = oci_connect("scott", "tiger", $db);
$c2 = oci_new_connect("scott", "tiger", $db);

function create_table($conn)
{
  $stmt = oci_parse($conn, "create table scott.hallo (test
varchar2(64))");
  oci_execute($stmt);
  echo $conn . " created table\n\n";
}

function drop_table($conn)
{
  $stmt = oci_parse($conn, "drop table scott.hallo");
  oci_execute($stmt);
  echo $conn . " dropped table\n\n";
}

function insert_data($conn)
{
  $stmt = oci_parse($conn, "insert into scott.hallo
           values('$conn' || ' ' || to_char(sysdate,'DD-MON-YY HH24:MI:SS'))");
  oci_execute($stmt, OCI_DEFAULT);
  echo $conn . " inserted hallo\n\n";
}

function delete_data($conn)
{
  $stmt = oci_parse($conn, "delete from scott.hallo");
  oci_execute($stmt, OCI_DEFAULT);
  echo $conn . " deleted hallo\n\n";
}

function commit($conn)
{
  oci_commit($conn);
  echo $conn . " committed\n\n";
}

function rollback($conn)
{
  oci_rollback($conn);
  echo $conn . " rollback\n\n";
}

function select_data($conn)
{
  $stmt = oci_parse($conn, "select * from scott.hallo");
  oci_execute($stmt, OCI_DEFAULT);
  echo $conn . "----selecting\n\n";
  while (oci_fetch($stmt)) {
    echo $conn . " <" . oci_result($stmt, "TEST") . ">\n\n";
  }
  echo $conn . "----done\n\n";
}

create_table($c1);
insert_data($c1);

select_data($c1);
select_data($c2);

rollback($c1);

select_data($c1);
select_data($c2);

insert_data($c2);
```

oci_new_connect() returns **FALSE** on error.

> **Nota:** In PHP versions before 5.0.0 you must use **ocinlogon()** instead. This name still can be used, it was left as alias of **oci_new_connect()** for downwards compatability. This, however, is deprecated and not recommended.

See also **oci_connect()** and **oci_pconnect()**.

oci_new_cursor

(PHP 5)

oci_new_cursor -- Allocates and returns a new cursor (statement handle)

Description

resource **oci_new_cursor** (resource connection)

oci_new_cursor() allocates a new statement handle on the specified connection.

Ejemplo 1. Using REF CURSOR in an Oracle's stored procedure

```
<?php
// suppose your stored procedure info.output returns a ref cursor in :data

$conn = oci_connect("scott", "tiger");
$curs = oci_new_cursor($conn);
$stmt = oci_parse($conn, "begin info.output(:data); end;");

oci_bind_by_name($stmt, "data", $curs, -1, OCI_B_CURSOR);
oci_execute($stmt);
oci_execute($curs);

while ($data = oci_fetch_row($curs)) {
    var_dump($data);
}

oci_free_statement($stmt);
oci_free_statement($curs);
oci_close($conn);
?>
```

Ejemplo 2. Using REF CURSOR in an Oracle's select statement

```
<?php
echo "<html><body>";
$conn = oci_connect("scott", "tiger");
$count_cursor = "CURSOR(select count(empno) num_emps from emp " .
                "where emp.deptno = dept.deptno) as EMPCNT from dept";
$stmt = oci_parse($conn, "select deptno,dname,$count_cursor");

oci_execute($stmt);
echo "<table border=\"1\">";
echo "<tr>";
echo "<th>DEPT NAME</th>";
echo "<th>DEPT #</th>";
echo "<th># EMPLOYEES</th>";
echo "</tr>";

while ($data = oci_fetch_assoc($stmt)) {
    echo "<tr>";
    $dname  = $data["DNAME"];
    $deptno = $data["DEPTNO"];
    echo "<td>$dname</td>";
    echo "<td>$deptno</td>";
    oci_execute($data["EMPCNT"]);
    while ($subdata = oci_fetch_assoc($data["EMPCNT"])) {
        $num_emps = $subdata["NUM_EMPS"];
        echo  "<td>$num_emps</td>";
    }
    echo "</tr>";
}
echo "</table>";
echo "</body></html>";
oci_free_statement($stmt);
oci_close($conn);
?>
```

oci_new_cursor() returns **FALSE** on error.

> **Nota:** In PHP versions before 5.0.0 you must use **ocinewcursor()** instead. This name still can be used, it was left as alias of **oci_new_cursor()** for downwards compatability. This, however, is deprecated and not recommended.

oci_new_descriptor

(PHP 5)

oci_new_descriptor -- Initializes a new empty LOB or FILE descriptor

Description

OCI-Lob **oci_new_descriptor** (resource connection [, int type])

oci_new_descriptor() allocates resources to hold descriptor or LOB locator. Valid values for *type* are: **OCI_D_FILE**, **OCI_D_LOB** and **OCI_D_ROWID**.

Ejemplo 1. oci_new_descriptor() example

```php
<?php
    /* This script is designed to be called from a HTML form.
     * It expects $user, $password, $table, $where, and $commitsize
     * to be passed in from the form.  The script then deletes
     * the selected rows using the ROWID and commits after each
     * set of $commitsize rows. (Use with care, there is no rollback)
     */
    $conn = oci_connect($user, $password);
    $stmt = oci_parse($conn, "select rowid from $table $where");
    $rowid = oci_new_descriptor($conn, OCI_D_ROWID);
    oci_define_by_name($stmt, "ROWID", $rowid);
    oci_execute($stmt);
    while (oci_fetch($stmt)) {
       $nrows = oci_num_rows($stmt);
       $delete = oci_parse($conn, "delete from $table where ROWID = :rid");
       oci_bind_by_name($delete, ":rid", $rowid, -1, OCI_B_ROWID);
       oci_execute($delete);
       echo "$nrows\n";
       if (($nrows % $commitsize) == 0) {
           oci_commit($conn);
       }
    }
    $nrows = oci_num_rows($stmt);
    echo "$nrows deleted...\n";
    oci_free_statement($stmt);
    oci_close($conn);
?>
```

```php
<?php
    /* This script demonstrates file upload to LOB columns
     * The formfield used for this example looks like this
     * <form action="upload.php" method="post" enctype="multipart/form-data">
     * <input type="file" name="lob_upload" />
     * ...
     */
  if (!isset($lob_upload) || $lob_upload == 'none'){
?>
<form action="upload.php" method="post" enctype="multipart/form-data">
Upload file: <input type="file" name="lob_upload" /><br />
<input type="submit" value="Upload" /> - <input type="reset" value="Reset" />
</form>
<?php
  } else {

     // $lob_upload contains the temporary filename of the uploaded file

     // see also the features section on file upload,
     // if you would like to use secure uploads

     $conn = oci_connect($user, $password);
     $lob = oci_new_descriptor($conn, OCI_D_LOB);
     $stmt = oci_parse($conn, "insert into $table (id, the_blob)
               values(my_seq.NEXTVAL, EMPTY_BLOB()) returning the_blob into :the_blob");
     oci_bind_by_name($stmt, ':the_blob', $lob, -1, OCI_B_BLOB);
     oci_execute($stmt, OCI_DEFAULT);
     if ($lob->savefile($lob_upload)){
        oci_commit($conn);
        echo "Blob successfully uploaded\n";
     }else{
        echo "Couldn't upload Blob\n";
     }
     oci_free_descriptor($lob);
     oci_free_statement($stmt);
     oci_close($conn);
  }
?>
```

Ejemplo 2. oci_new_descriptor() example

```
<?php
    /* Calling PL/SQL stored procedures which contain clobs as input
     * parameters (PHP 4 >= 4.0.6).
     * Example PL/SQL stored procedure signature is:
     *
     * PROCEDURE save_data
     *   Argument Name                  Type                      In/Out Default?
     *   ------------------------------ ------------------------- ------ --------
     *   KEY                            NUMBER(38)                IN
     *   DATA                           CLOB                      IN
     *
     */

    $conn = oci_connect($user, $password);
    $stmt = oci_parse($conn, "begin save_data(:key, :data); end;");
    $clob = oci_new_descriptor($conn, OCI_D_LOB);
    oci_bind_by_name($stmt, ':key', $key);
    oci_bind_by_name($stmt, ':data', $clob, -1, OCI_B_CLOB);
    $clob->write($data);
    oci_execute($stmt, OCI_DEFAULT);
    oci_commit($conn);
    $clob->free();
    oci_free_statement($stmt);
?>
```

oci_new_descriptor() returns **`FALSE`** on error.

> **Nota:** In PHP versions before 5.0.0 you must use **ocinewdescriptor()** instead. This name still can be used, it was left as alias of **oci_new_descriptor()** for downwards compatability. This, however, is deprecated and not recommended.

oci_num_fields

(PHP 5)

oci_num_fields -- Returns the number of result columns in a statement

Description

int **oci_num_fields** (resource statement)

oci_num_fields() returns the number of columns in the *statement*.

Ejemplo 1. oci_num_fields() example

```
<?php
    echo "<pre>\n";
    $conn = oci_connect("scott", "tiger");
    $stmt = oci_parse($conn, "select * from emp");

    oci_execute($stmt);

    while (oci_fetch($stmt)) {
        echo "\n";
        $ncols = oci_num_fields($stmt);
        for ($i = 1; $i <= $ncols; $i++) {
            $column_name  = oci_field_name($stmt, $i);
            $column_value = oci_result($stmt, $i);
            echo $column_name . ': ' . $column_value . "\n";
        }
        echo "\n";
    }

    oci_free_statement($stmt);
    oci_close($conn);

    echo "</pre>";
?>
```

oci_num_fields() returns **FALSE** on error.

> **Nota:** In PHP versions before 5.0.0 you must use **ocinumcols()** instead. This name still can be used, it was left as alias of **oci_num_fields()** for downwards compatability. This, however, is deprecated and not recommended.

oci_num_rows

(PHP 5)

oci_num_rows -- Returns number of rows affected during statement execution

Description

int **oci_num_rows** (resource stmt)

oci_num_rows() returns number of rows affected during statement execution.

> **Nota:** This function *does not* return number of rows selected! For SELECT statements this function will return the number of rows, that were fetched to the buffer with **oci_fetch*()** functions.

Ejemplo 1. oci_num_rows() example

```
<?php
    echo "<pre>";
    $conn = oci_connect("scott", "tiger");

    $stmt = oci_parse($conn, "create table emp2 as select * from emp");
    oci_execute($stmt);
    echo oci_num_rows($stmt) . " rows inserted.<br />";
    oci_free_statement($stmt);

    $stmt = oci_parse($conn, "delete from emp2");
    oci_execute($stmt, OCI_DEFAULT);
    echo oci_num_rows($stmt) . " rows deleted.<br />";
    oci_commit($conn);
    oci_free_statement($stmt);

    $stmt = oci_parse($conn, "drop table emp2");
    oci_execute($stmt);
    oci_free_statement($stmt);

    oci_close($conn);
    echo "</pre>";
?>
```

oci_num_rows() returns **FALSE** on error.

> **Nota:** In PHP versions before 5.0.0 you must use **ocirowcount()** instead. This name still can be used, it was left as alias of **oci_num_rows()** for downwards compatability. This, however, is deprecated and not recommended.

oci_parse

(PHP 5)

oci_parse -- Prepares Oracle statement for execution

Description

resource **oci_parse** (resource connection, string query)

oci_parse() prepares the *query* using *connection* and returns the statement identifier, which can be used with **oci_bind_by_name()**, **oci_execute()** and other functions.

> **Nota:** This function *does not* validate *query*. The only way to find out if *query* is valid SQL or PL/SQL statement - is to execute it.

oci_parse() returns **FALSE** on error.

> **Nota:** In PHP versions before 5.0.0 you must use **ociparse()** instead. This name still can be used, it was left as alias of **oci_parse()** for downwards compatability. This, however, is deprecated and not recommended.

oci_password_change

(PHP 5)

oci_password_change -- Changes password of Oracle's user

Description

bool **oci_password_change** (resource connection, string username, string old_password, string new_password)

Changes password for user with *username*. Parameters *old_password* and *new_password* should indicate old and new passwords respectively.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

> **Nota:** In PHP versions before 5.0.0 you must use **ocipasswordchange()** instead. This name still can be used, it was left as alias of **oci_password_change()** for downwards compatability. This, however, is deprecated and not recommended.

oci_pconnect

(PHP 5)

oci_pconnect -- Connect to an Oracle database using a persistent connection

Description

resource **oci_pconnect** (string username, string password [, string db [, string charset]])

oci_pconnect() creates a new persistent connection to an Oracle server and logs on. The optional third parameter can either contain the name of the local Oracle instance or the name of the entry in `tnsnames.ora`. If the third parameter is not specified, PHP uses environment variables *ORACLE_SID* and *TWO_TASK* to determine the name of local Oracle instance and location of `tnsnames.ora` accordingly.

Si usais Oracle server v.9.2 ó superior, podeis definir el parametro *charset*, el cual será utilizado en la nueva conexión. Si utilizais una version < 9.2, este parámetro será ignorado y la variable de entorno NLS_LANG será usada en su lugar.

oci_pconnect() returns connection identifier or **FALSE** on error.

> **Nota:** Note, that these kind of links only work if you are using a module version of PHP. See the Persistent Database Connections section for more information.

> **Nota:** In PHP versions before 5.0.0 you must use **ociplogon()** instead. This name still can be used, it was left as alias of **oci_pconnect()** for downwards compatability. This, however, is deprecated and not recommended.

See also **oci_connect()** and **oci_new_connect()**.

oci_result

(PHP 5)

oci_result -- Returns field's value from the fetched row

Description

mixed **oci_result** (resource statement, mixed field)

oci_result() returns the data from the field *field* in the current row, fetched by **oci_fetch()**.
oci_result() returns everything as strings except for abstract types (ROWIDs, LOBs and FILEs).
oci_result() returns **FALSE** on error.

You can either use the column number (1-based) or the column name (in uppercase) for the **field()** parameter.

> **Nota:** In PHP versions before 5.0.0 you must use **ociresult()** instead. This name still can be used, it was left as alias of **oci_result()** for downwards compatability. This, however, is deprecated and not recommended.

See also **oci_fetch_array()**, **oci_fetch_assoc()**, **oci_fetch_object()**, **oci_fetch_row()** and **oci_fetch_all()**.

oci_rollback

(PHP 5)

oci_rollback -- Rolls back outstanding transaction

Description

bool **oci_rollback** (resource connection)

oci_rollback() rolls back all outstanding statements for Oracle connection *connection*.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

> **Nota:** In PHP versions before 5.0.0 you must use **ocirollback()** instead. This name still can be used, it was left as alias of **oci_rollback()** for downwards compatability. This, however, is deprecated and not recommended.

See also **oci_commit()**.

oci_server_version

(PHP 5)

oci_server_version -- Returns server version

Description

string **oci_server_version** (resource connection)

Returns a string with version information of the Oracle server, which uses connection *connection* or returns **FALSE** on error.

Ejemplo 1. oci_server_version() example

```
<?php
    $conn = oci_connect("scott", "tiger");
    echo "Server Version: " . oci_server_version($conn);
    oci_close($conn);
?>
```

> **Nota:** In PHP versions before 5.0.0 you must use **ociserverversion()** instead. This name still can be used, it was left as alias of **oci_server_version()** for downwards compatability. This, however, is deprecated and not recommended.

oci_set_prefetch

(PHP 5)

oci_set_prefetch -- Sets number of rows to be prefetched

Description

bool **oci_set_prefetch** (resource statement [, int rows])

Sets the number of rows to be prefetched after successful call to **oci_execute()**. The default value for *rows* is 1.

> **Nota:** In PHP versions before 5.0.0 you must use **ocisetprefetch()** instead. This name still can be used, it was left as alias of **oci_set_prefetch()** for downwards compatability. This, however, is deprecated and not recommended.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

oci_statement_type

(PHP 5)

oci_statement_type -- Returns the type of an OCI statement

Description

string **oci_statement_type** (resource statement)

oci_statement_type() returns the query type of statement *statement* as one of the following values:

1. *SELECT*
2. *UPDATE*
3. *DELETE*
4. *INSERT*
5. *CREATE*

6. *DROP*

7. *ALTER*

8. *BEGIN*

9. *DECLARE*

10. *UNKNOWN*

Parameter *statement* is a valid OCI statement identifier, returned from **oci_parse()**.

Ejemplo 1. oci_statement_type() example

```
<?php
    $conn = oci_connect("scott", "tiger");
    $sql  = "delete from emp where deptno = 10";

    $stmt = oci_parse($conn, $sql);
    if (oci_statement_type($stmt) == "DELETE") {
        die("You are not allowed to delete from this table<br />");
    }

    oci_close($conn);
?>
```

oci_statement_type() returns **FALSE** on error.

> **Nota:** In PHP versions before 5.0.0 you must use **ocistatementtype()** instead. This name still can be used, it was left as alias of **oci_statement_type()** for downwards compatability. This, however, is deprecated and not recommended.

OCIBindByName

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

OCIBindByName -- Enlaza una variable PHP a un Placeholder de Oracle

Descripción

int **OCIBindByName** (int stmt, string ph_name, mixed & variable, int length [, int type])

OCIBindByName() enlaza la variable PHP *variable* a un placeholder de Oracle *ph_name*. Si esta será usada para entrada o salida se determinará en tiempo de ejecución, y sera resevado el espacio necesario de almacenamiento. El parámetro *length* establece el tamaño máximo del enlace. Si establece *length* a -1 **OCIBindByName()** usará el tamaño de la *variable* para establecer el tamaño máximo.

Si necesita enlazar tipos de datos abstractos (LOB/ROWID/BFILE) necesitará primero reservar la memoria con la función **OCINewDescriptor()**. *length* no se usa para tipos de datos abstractos y debería establecerse a -1. La variable *type* le informa a Oracle, que tipo de descriptor queremos usar. Los valores posibles son: OCI_B_FILE (Binary-File), OCI_B_CFILE (Character-File), OCI_B_CLOB (Character-LOB), OCI_B_BLOB (Binary-LOB) and OCI_B_ROWID (ROWID).

Ejemplo 1. OCIDefineByName

```
<?php
/* OCIBindByPos example thies@digicol.de (980221)

  inserts 3 resords into emp, and uses the ROWID for updating the
  records just after the insert.
*/

$conn = OCILogon("scott","tiger");

$stmt = OCIParse($conn,"insert into emp (empno, ename) ".
                       "values (:empno,:ename) ".
                       "returning ROWID into :rid");

$data = array(1111 => "Larry", 2222 => "Bill", 3333 => "Jim");

$rowid = OCINewDescriptor($conn,OCI_D_ROWID);

OCIBindByName($stmt,":empno",&$empno,32);
OCIBindByName($stmt,":ename",&$ename,32);
OCIBindByName($stmt,":rid",&$rowid,-1,OCI_B_ROWID);

$update = OCIParse($conn,"update emp set sal = :sal where ROWID = :rid");
OCIBindByName($update,":rid",&$rowid,-1,OCI_B_ROWID);
OCIBindByName($update,":sal",&$sal,32);

$sal = 10000;

while (list($empno,$ename) = each($data)) {
    OCIExecute($stmt);
    OCIExecute($update);
}

$rowid->free();

OCIFreeStatement($update);
OCIFreeStatement($stmt);

$stmt = OCIParse($conn,"select * from emp where empno in (1111,2222,3333)");
OCIExecute($stmt);
while (OCIFetchInto($stmt,&$arr,OCI_ASSOC)) {
    var_dump($arr);
}
OCIFreeStatement($stmt);

/* delete our "junk" from the emp table.... */
$stmt = OCIParse($conn,"delete from emp where empno in (1111,2222,3333)");
OCIExecute($stmt);
OCIFreeStatement($stmt);

OCILogoff($conn);
?>
```

ocicancel

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

ocicancel -- Cancel reading from cursor

Description

bool **ocicancel** (resource stmt)

If you do not want read more data from a cursor, then call **ocicancel()**.

> **Nota:** This function was renamed to **oci_cancel()** after PHP >= 5.0.0. For downward compatibility **ocicancel()** can also be used. This is deprecated, however.

ocicloselob

(no version information, might be only in CVS)

ocicloselob -- Closes lob descriptor

Description

bool **ocicloselob** (void)

> **Nota:** This function was renamed to **oci_lob_close()** after PHP >= 5.0.0. For downward compatibility **ocicloselob()** can also be used. This is deprecated, however.

ocicollappend

(PHP 4 >= 4.0.6, PHP 5)

ocicollappend -- Append an object to the collection

Description

bool **ocicollappend** (string value)

> **Nota:** This function was renamed to **oci_collection_append()** after PHP >= 5.0.0. For downward compatibility **ocicollappend()** can also be used. This is deprecated, however.

ocicollassign

(PHP 4 >= 4.0.6)

ocicollassign -- Assign a collection from another existing collection

Description

bool **ocicollassign** (OCI-Collection from)

> **Nota:** This function was renamed to **oci_collection_assign()** after PHP >= 5.0.0. For downward compatibility **ocicollassign()** can also be used. This is deprecated, however.

ocicollassignelem

(PHP 4 >= 4.0.6, PHP 5)

ocicollassignelem -- Assign element val to collection at index ndx

Description

bool **ocicollassignelem** (int ndx, string val)

> **Nota:** This function was renamed to **oci_collection_element_assign()** after PHP >= 5.0.0. For downward compatibility **ocicollassignelem()** can also be used. This is deprecated, however.

ocicollgetelem

(PHP 4 >= 4.0.6, PHP 5)

ocicollgetelem -- Retrieve the value at collection index ndx

Description

string **ocicollgetelem** (int ndx)

> **Nota:** This function was renamed to **oci_collection_element_get()** after PHP >= 5.0.0. For downward compatibility **ocicollgetelem()** can also be used. This is deprecated, however.

ocicollmax

(PHP 4 >= 4.0.6, PHP 5)

ocicollmax -- Gets the maximum number of elements in the collection

Description

int **ocicollmax** (void)

> **Nota:** This function was renamed to **oci_collection_max()** after PHP >= 5.0.0. For downward compatibility **ocicollmax()** can also be used. This is deprecated, however.

ocicollsize

(PHP 4 >= 4.0.6, PHP 5)

ocicollsize -- Return the size of a collection

Description

int **ocicollsize** (void)

> **Nota:** This function was renamed to **oci_collection_size()** after PHP >= 5.0.0. For downward compatibility **ocicollsize()** can also be used. This is deprecated, however.

ocicolltrim

(PHP 4 >= 4.0.6, PHP 5)

ocicolltrim -- Trim num elements from the end of a collection

Description

bool **ocicolltrim** (int num)

> **Nota:** This function was renamed to **oci_collection_trim()** after PHP >= 5.0.0. For downward compatibility **ocicolltrim()** can also be used. This is deprecated, however.

OCIColumnIsNULL

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

OCIColumnIsNULL -- comprueba si una una columna es **`NULL`**

Descripción

int **OCIColumnIsNULL** (int stmt, mixed column)

OCIColumnIsNULL() devuelve vedadero si la columna devuelta *column* en el resultset de la sentencia *stmt* es **`NULL`**. Puede usar el número de la columna (1-Based) o el nombre de la columa indicado por el parámetro *col*.

OCIColumnName

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

OCIColumnName -- Devuelve el nombre de una columna.

Descripción

string **OCIColumnName** (int stmt, int col)

OCIColumnName() Devuelve el nombre de la columna correspondiente al número de la columna (1-based) que es pasado.

Ejemplo 1. OCIColumnName

```
<?php
    print "<HTML><PRE>\n";
    $conn = OCILogon("scott", "tiger");
    $stmt = OCIParse($conn,"select * from emp");
    OCIExecute($stmt);
    print "<TABLE BORDER=\"1\">";
    print "<TR>";
    print "<TH>Name</TH>";
    print "<TH>Type</TH>";
    print "<TH>Length</TH>";
    print "</TR>";
    $ncols = OCINumCols($stmt);
    for ( $i = 1; $i <= $ncols; $i++ ) {
        $column_name  = OCIColumnName($stmt,$i);
        $column_type  = OCIColumnType($stmt,$i);
        $column_size  = OCIColumnSize($stmt,$i);
        print "<TR>";
        print "<TD>$column_name</TD>";
        print "<TD>$column_type</TD>";
        print "<TD>$column_size</TD>";
        print "</TR>";
    }
    OCIFreeStatement($stmt);
    OCILogoff($conn);
    print "</PRE>";
    print "</HTML>\n";
?>
```

Vea también **OCINumCols()**, **OCIColumnType()**, y **OCIColumnSize()**.

ocicolumnprecision

(PHP 4 , PHP 5)

ocicolumnprecision -- Tell the precision of a column

Description

int **ocicolumnprecision** (resource stmt, int col)

Returns precision of the field with *col* index (1-based).

For FLOAT columns precision is nonzero and scale is -127. If precision is 0, then columnt is NUMBER. Else it's NUMBER(precision, scale).

> **Nota:** This function was renamed to **oci_field_precision()** after PHP >= 5.0.0. For downward compatibility **ocicolumnprecision()** can also be used. This is deprecated, however.

ocicolumnscale

(PHP 4 , PHP 5)

ocicolumnscale -- Tell the scale of a column

Description

int **ocicolumnscale** (resource stmt, int col)

Nota: This function was renamed to **oci_field_scale()** after PHP >= 5.0.0. For downward compatibility **ocicolumnscale()** can also be used. This is deprecated, however.

OCIColumnSize

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

OCIColumnSize -- devuelve el tamaño de la columna

Descripción

int **OCIColumnSize** (int stmt, mixed column)

OCIColumnSize() devuelve el tamaño de la columna indicada por Oracle Puede utilizar el número de la columna (1-Based) o el nombre indicado en el parámetro *col*.

Ejemplo 1. OCIColumnSize

```
<?php
    print "<HTML><PRE>\n";
    $conn = OCILogon("scott", "tiger");
    $stmt = OCIParse($conn,"select * from emp");
    OCIExecute($stmt);
    print "<TABLE BORDER=\"1\">";
    print "<TR>";
    print "<TH>Name</TH>";
    print "<TH>Type</TH>";
    print "<TH>Length</TH>";
    print "</TR>";
    $ncols = OCINumCols($stmt);
    for ( $i = 1; $i <= $ncols; $i++ ) {
        $column_name  = OCIColumnName($stmt,$i);
        $column_type  = OCIColumnType($stmt,$i);
        $column_size  = OCIColumnSize($stmt,$i);
        print "<TR>";
        print "<TD>$column_name</TD>";
        print "<TD>$column_type</TD>";
        print "<TD>$column_size</TD>";
        print "</TR>";
    }
    print "</TABLE>";
    OCIFreeStatement($stmt);
    OCILogoff($conn);
    print "</PRE>";
    print "</HTML>\n";
?>
```

Vea también **OCINumCols()**, **OCIColumnName()**, y **OCIColumnSize()**.

OCIColumnType

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

OCIColumnType -- Devuelve el tipo de dato de una columna.

Descripción

mixed **OCIColumnType** (int stmt, int col)

OCIColumnType() devuelve el tipo de dato de una columna correspondiente al número de la columna (1-based) que es pasado.

Ejemplo 1. OCIColumnType

```
<?php
    print "<HTML><PRE>\n";
    $conn = OCILogon("scott", "tiger");
    $stmt = OCIParse($conn,"select * from emp");
    OCIExecute($stmt);
    print "<TABLE BORDER=\"1\">";
    print "<TR>";
    print "<TH>Name</TH>";
    print "<TH>Type</TH>";
    print "<TH>Length</TH>";
    print "</TR>";
    $ncols = OCINumCols($stmt);
    for ( $i = 1; $i <= $ncols; $i++ ) {
        $column_name  = OCIColumnName($stmt,$i);
        $column_type  = OCIColumnType($stmt,$i);
        $column_size  = OCIColumnSize($stmt,$i);
        print "<TR>";
        print "<TD>$column_name</TD>";
        print "<TD>$column_type</TD>";
        print "<TD>$column_size</TD>";
        print "</TR>";
    }
    OCIFreeStatement($stmt);
    OCILogoff($conn);
    print "</PRE>";
    print "</HTML>\n";
?>
```

Vea también **OCINumCols()**, **OCIColumnName()**, y **OCIColumnSize()**.

ocicolumntyperaw

(PHP 4 , PHP 5)

ocicolumntyperaw -- Tell the raw oracle data type of a column

Description

int **ocicolumntyperaw** (resource stmt, int col)

ocicolumntyperaw() returns Oracle's raw data type of the field.

> **Nota:** This function was renamed to **oci_field_type_raw()** after PHP >= 5.0.0. For downward compatibility **ocicolumntyperaw()** can also be used. This is deprecated, however.

OCICommit

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

OCICommit -- Confirma transacciones pendientes

Descripción

int **OCICommit** (int connection)

OCICommit() confirma todas las sentencias pendientes para la conexión con Oracle *connection*.

OCIDefineByName

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

OCIDefineByName -- Usa una variable de PHP para el define-step durante una sentencia SELECT

Descripción

int **OCIDefineByName** (int stmt, string Column-Name, mixed & variable [, int type])

OCIDefineByName() busca el valor de las Columnas-SQL dentro de variables PHP definidas por el usuario. Cuidado que Oracle nombra todas las columnas en MAYUSCULAS, mientras que en su select puede usar también minúsculas write lower-case. **OCIDefineByName()** espera que *Column-Name* esté en mayúsculas. Si define una variable que no existe en la sentecia SELECT, no se producirá ningún error.

Si necesita definir un tipo de dato abstracto (LOB/ROWID/BFILE) tendrá que alojarlo primero usando la función **OCINewDescriptor()** function. Vea también la función **OCIBindByName()**.

Ejemplo 1. OCIDefineByName

```
<?php
/* OCIDefineByPos example thies@digicol.de (980219) */

$conn = OCILogon("scott","tiger");

$stmt = OCIParse($conn,"select empno, ename from emp");

/* la definición DEBE hacerse ANTES del ociexecute! */

OCIDefineByName($stmt,"EMPNO",&$empno);
OCIDefineByName($stmt,"ENAME",&$ename);

OCIExecute($stmt);

while (OCIFetch($stmt)) {
    echo "empno:".$empno."\n";
    echo "ename:".$ename."\n";
}

OCIFreeStatement($stmt);
OCILogoff($conn);
?>
```

OCIError

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

OCIError -- Devuelve el último error de stmt|conn|global. Si no ocurre ningún error devuelve falso.

Descripción

array **OCIError** ([int stmt|conn|global])

OCIError() devuelve el último error encontrado. Si el parámetro opcional *stmt|conn|global* no es usado, es devuelto el último error encontrado. Si no se encuentra ningún error, **OCIError()** devuelve falso. **OCIError()** devuelve el error como un array asociativo. En este array, *code* consiste en el código de error de Oracle y *message* en la cadena de descripción del error.

OCIExecute

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

OCIExecute -- Ejecuta una sentencia

Descripción

int **OCIExecute** (int statement [, int mode])

OCIExecute() ejecuta una sentencia previamente analizada. (see **OCIParse()**). El parámetro opcional *mode* le permite especificar el modo de ejecución (default is OCI_COMMIT_ON_SUCCESS). Si no desea que las sentencias se confirmen automaticamente, especifique OCI_DEFAULT como su modo.

OCIFetch

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

OCIFetch -- Busca la siguiente fila en el result-buffer

Descripción

int **OCIFetch** (int statement)

OCIFetch() Busca la siguiente fila (para sentencias SELECT) dentro del result-buffer interno.

OCIFetchInto

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

OCIFetchInto -- Busca la siguiente fila dentro del result-array

Descripción

int **OCIFetchInto** (int stmt, array & result [, int mode])

OCIFetchInto() busca la siguiente fila (for SELECT statements) dentro del array *result*. **OCIFetchInto()** sobreescribirá el contenido previo de *result*. Por defecto *result* contendrá un array basado en todas las columnas que no son **NULL**.

El parámetro *mode* le permite cambiar el comportamineto por defecto. Puede especificar más de una flag simplemente añadiendolas (ej. OCI_ASSOC+OCI_RETURN_NULLS). Las flags son:

OCI_ASSOC Devuelve un array asociativo.
OCI_NUM Devuelve un array numerado empezando en 1. (POR DEFECTO)
OCI_RETURN_NULLS Devuelve columnas vacias.
OCI_RETURN_LOBS Devuelve el valor de un LOB en vez de el descriptor.

OCIFetchStatement

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

OCIFetchStatement -- Busca todas la filas de un resultset dentro de un array.

Descripción

int **OCIFetchStatement** (int stmt, array & variable)

OCIFetchStatement() busca todas las filas de un resultset dentro de un array definido por el usuario. **OCIFetchStatement()** devuelve el numero de filas buscadas.

Ejemplo 1. OCIFetchStatement

```
<?php
/* OCIFetchStatement example mbritton@verinet.com (990624) */

$conn = OCILogon("scott","tiger");

$stmt = OCIParse($conn,"select * from emp");

OCIExecute($stmt);

$nrows = OCIFetchStatement($stmt,$results);
if ( $nrows > 0 ) {
   print "<TABLE BORDER=\"1\">\n";
   print "<TR>\n";
   while ( list( $key, $val ) = each( $results ) ) {
      print "<TH>$key</TH>\n";
   }
   print "</TR>\n";

   for ( $i = 0; $i < $nrows; $i++ ) {
      reset($results);
      print "<TR>\n";
      while ( $column = each($results) ) {
         $data = $column['value'];
         print "<TD>$data[$i]</TD>\n";
      }
      print "</TR>\n";
   }
   print "</TABLE>\n";
} else {
   echo "No data found<BR>\n";
}
print "$nrows Records Selected<BR>\n";

OCIFreeStatement($stmt);
OCILogoff($conn);

?>
```

ocifreecollection

(PHP 4 >= 4.1.0, PHP 5)

ocifreecollection -- Deletes collection object

Description

bool **ocifreecollection** (void)

> **Nota:** This function was renamed to **oci_free_collection()** after PHP >= 5.0.0. For downward compatibility **ocifreecollection()** can also be used. This is deprecated, however.

OCIFreeCursor

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

OCIFreeCursor -- Libera todos los recursos asociados con cursor.

Descripción

int **OCIFreeCursor** (int stmt)

OCIFreeCursor() devuelve cierto si la operacion se lleva a cabo, o falso en caso contrario.

ocifreedesc

(PHP 4 , PHP 5)

ocifreedesc -- Deletes a large object descriptor

Description

bool **ocifreedesc** (void)

ocifreedesc() deletes a large object descriptor. Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

> **Nota:** This function was renamed to oci_free_descriptor() after PHP >= 5.0.0. For downward compatibility **ocifreedesc()** can also be used. This is deprecated, however.

OCIFreeStatement

(PHP 3>= 3.0.5, PHP 4 , PHP 5)

OCIFreeStatement -- Libera todos los recursos asociados con una sentencia.

Descripción

int **OCIFreeStatement** (int stmt)

OCIFreeStatement() devuelve cierto si la operacion se lleva a cabo, o falso en caso contrario.

lob->getBuffering

(no version information, might be only in CVS)

lob->getBuffering -- Returns current state of buffering for large object

Description

bool **lob->getBuffering** (void)

Returns **`FALSE`** if buffering for the large object is off and **`TRUE`** if buffering is used.

See also **ocisetbufferinglob()**.

OCIInternalDebug

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

OCIInternalDebug -- Habilita o deshabilita la salida del depurador interno. Por defecto este está deshabiltado

Descripción

void **OCIInternalDebug** (int onoff)

OCIInternalDebug() habilita la salida del depurador interno. Asigne 0 a *onoff* para deshabilitar la salida y 1 para habilitarla.

ociloadlob

(PHP 4 , PHP 5)

ociloadlob -- Loads a large object

Description

string **ociloadlob** (void)

> **Nota:** This function was renamed to **oci_lob_load()** after PHP >= 5.0.0. For downward compatibility **ociloadlob()** can also be used. This is deprecated, however.

OCILogOff

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

OCILogOff -- Termina la conexion con Oracle

Descripción

int **OCILogOff** (int connection)

OCILogOff() cierra una conexión con Oracle.

ocilogon

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

ocilogon -- Establece una conexión con Oracle

Descripción

resource **ocilogon** (string nombre_usuario, string contrasenya [, string bd [, string juego_caracteres]])

ocilogon() devuelve un identificador de conexión necesario para la mayoría de las demás llamadas OCI. El tercer parámetro opcional puede contener el nombre de la instancia local a Oracle o el nombre de una entrada en tnsnames.ora con la cual desea conectarse. Si el tercer parámetro opcional no se especifica, PHP usa las variables de entorno ORACLE_SID (instancia de Oracle) o TWO_TASK (tnsnames.ora) para determinar a cuál base de datos conectarse.

Si usais Oracle server v.9.2 ó superior, podeis definir el parametro *charset*, el cual será utilizado en la nueva conexión. Si utilizais una version < 9.2, este parámetro será ignorado y la variable de entorno NLS_LANG será usada en su lugar.

Las conexiones son compartidas al nivel de página cuando se use **ocilogon()**. Esto quiere decir que los procesos "commit" y "rollback" se aplican a todas las transacciones abiertas en la página, incluso si usted ha creado conexiones múltiples.

Este ejemplo demuestra cómo son compartidas las conexiones.
Ejemplo 1. Ejemplo de ocilogon()

```php
<?php
echo "<pre>";
$bd = "";

$c1 = ocilogon("scott", "tiger", $bd);
$c2 = ocilogon("scott", "tiger", $bd);

function crear_tabla($con)
{
  $sentencia = ociparse($con, "create table scott.hallo (test varchar2(64))");
  ociexecute($sentencia);
  echo $con . " tabla creada\n\n";
}

function abandonar_tabla($con)
{
  $sentencia = ociparse($con, "drop table scott.hallo");
  ociexecute($sentencia);
  echo $con . " tabla abandonada\n\n";
}

function insertar_datos($con)
{
  $sentencia = ociparse($con, "insert into scott.hallo
           values('$con' || ' ' || to_char(sysdate,'DD-MON-YY HH24:MI:SS'))");
  ociexecute($sentencia, OCI_DEFAULT);
  echo $con . " datos insertados en hallo\n\n";
}

function eliminar_datos($con)
{
  $sentencia = ociparse($con, "delete from scott.hallo");
  ociexecute($sentencia, OCI_DEFAULT);
  echo $con . " datos eliminados de hallo\n\n";
}

function commit($con)
{
  ocicommit($con);
  echo $con . " committed\n\n";
}

function rollback($con)
{
  ocirollback($con);
  echo $con . " rollback\n\n";
}

function seleccionar_datos($con)
{
  $sentencia = ociparse($con, "select * from scott.hallo");
  ociexecute($sentencia, OCI_DEFAULT);
  echo $con."----seleccionando\n\n";
  while (ocifetch($sentencia)) {
    echo $con . " [" . ociresult($sentencia, "TEST") . "]\n\n";
  }
  echo $con . "----listo\n\n";
}

crear_tabla($c1);
insertar_datos($c1);   // Insertar una fila usando c1
insertar_datos($c2);   // Insertar una fila usando c2

seleccionar_datos($c1);   // Son devueltos los resultados de ambas inserciones
seleccionar_datos($c2);

rollback($c1);      // Rollback usando c1

seleccionar_datos($c1);   // Ambas inserciones han sido devueltas
seleccionar_datos($c2);

insertar_datos($c2);   // Insertar una fila usando c2
```

Vea también **ociplogon()** y **ocinlogon()**.

ocinewcollection

(PHP 4 >= 4.0.6, PHP 5)

ocinewcollection -- Initialize a new collection

Description

OCI-Collection **ocinewcollection** (resource connection, string tdo [, string schema])

> **Nota:** This function was renamed to **oci_new_collection()** after PHP >= 5.0.0. For downward compatibility **ocinewcollection()** can also be used. This is deprecated, however.

OCINewCursor

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

OCINewCursor -- devuelve un cursor nuevo (Statement-Handle) - use esto para enlazar ref-cursors!

Descripción

int **OCINewCursor** (int conn)

OCINewCursor() allocates a new statement handle on the specified connection.

Ejemplo 1. Usando un REF CURSOR de un procedimiento almacenado

```
<?php
// suppose your stored procedure info.output returns a ref cursor in :data

$conn = OCILogon("scott","tiger");
$curs = OCINewCursor($conn);
$stmt = OCIParse($conn,"begin info.output(:data); end;");

ocibindbyname($stmt,"data",&$curs,-1,OCI_B_CURSOR);
ociexecute($stmt);
ociexecute($curs);

while (OCIFetchInto($curs,&$data)) {
    var_dump($data);
}

OCIFreeCursor($stmt);
OCIFreeStatement($curs);
OCILogoff($conn);
?>
```

Ejemplo 2. Usando un REF CURSOR en una sentencia select

```
<?php
print "<HTML><BODY>";
$conn = OCILogon("scott","tiger");
$count_cursor = "CURSOR(select count(empno) num_emps from emp " .
                "where emp.deptno = dept.deptno) as EMPCNT from dept";
$stmt = OCIParse($conn,"select deptno,dname,$count_cursor");

ociexecute($stmt);
print "<TABLE BORDER=\"1\">";
print "<TR>";
print "<TH>DEPT NAME</TH>";
print "<TH>DEPT #</TH>";
print "<TH># EMPLOYEES</TH>";
print "</TR>";

while (OCIFetchInto($stmt,&$data,OCI_ASSOC)) {
    print "<TR>";
    $dname  = $data["DNAME"];
    $deptno = $data["DEPTNO"];
    print "<TD>$dname</TD>";
    print "<TD>$deptno</TD>";
    ociexecute($data[ "EMPCNT" ]);
    while (OCIFetchInto($data[ "EMPCNT" ],&$subdata,OCI_ASSOC)) {
        $num_emps = $subdata["NUM_EMPS"];
        print  "<TD>$num_emps</TD>";
    }
    print "</TR>";
}
print "</TABLE>";
print "</BODY></HTML>";
OCIFreeStatement($stmt);
OCILogoff($conn);
?>
```

OCINewDescriptor

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

OCINewDescriptor -- Inicializa un nuevo descriptor vacío LOB/FILE (LOB por defecto)

Descripción

string **OCINewDescriptor** (int connection [, int type])

OCINewDescriptor() Reserva espacio para mantener descriptores o localizadores LOB. Los valores válidos para el tipo *type* son OCI_D_FILE, OCI_D_LOB, OCI_D_ROWID. Para descriptores LOB, los métodos load, save, y savefile están asociados con el descriptor, para BFILE sólo el método load existe. Vea el segundo ejemplo.

Ejemplo 1. OCINewDescriptor

```php
<?php
    /* This script is designed to be called from a HTML form.
     * It expects $user, $password, $table, $where, and $commitsize
     * to be passed in from the form.  The script then deletes
     * the selected rows using the ROWID and commits after each
     * set of $commitsize rows. (Use with care, there is no rollback)
     */
    $conn = OCILogon($user, $password);
    $stmt = OCIParse($conn,"select rowid from $table $where");
    $rowid = OCINewDescriptor($conn,OCI_D_ROWID);
    OCIDefineByName($stmt,"ROWID",&$rowid);
    OCIExecute($stmt);
    while ( OCIFetch($stmt) ) {
       $nrows = OCIRowCount($stmt);
       $delete = OCIParse($conn,"delete from $table where ROWID = :rid");
       OCIBindByName($delete,":rid",&$rowid,-1,OCI_B_ROWID);
       OCIExecute($delete);
       print "$nrows\n";
       if ( ($nrows % $commitsize) == 0 ) {
           OCICommit($conn);
       }
    }
    $nrows = OCIRowCount($stmt);
    print "$nrows deleted...\n";
    OCIFreeStatement($stmt);
    OCILogoff($conn);
?>
```

```php
<?php
    /* This script demonstrates file upload to LOB columns
     * The formfield used for this example looks like this
     * <form action="upload.php3" method="post" enctype="multipart/form-data">
     * <input type="file" name="lob_upload">
     * ...
     */
  if(!isset($lob_upload) || $lob_upload == 'none'){
?>
<form action="upload.php3" method="post" enctype="multipart/form-data">
Upload file: <input type="file" name="lob_upload"><br>
<input type="submit" value="Upload"> - <input type="reset">
</form>
<?php
  } else {
     // $lob_upload contains the temporary filename of the uploaded file
     $conn = OCILogon($user, $password);
     $lob = OCINewDescriptor($conn, OCI_D_LOB);
     $stmt = OCIParse($conn,"insert into $table (id, the_blob)
                values(my_seq.NEXTVAL, EMPTY_BLOB()) returning the_blob into :the_blob");
     OCIBindByName($stmt, ':the_blob', &$lob, -1, OCI_B_BLOB);
     OCIExecute($stmt);
     if($lob->savefile($lob_upload)){
        OCICommit($conn);
        echo "Blob successfully uploaded\n";
     }else{
        echo "Couldn't upload Blob\n";
     }
     OCIFreeDescriptor($lob);
     OCIFreeStatement($stmt);
     OCILogoff($conn);
  }
?>
```

ocinlogon

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

ocinlogon -- Establece una nueva conexión con Oracle

Descripción

resource **ocinlogon** (string nombre_usuario, string contrasenya [, string bd [, string juego_caracteres]])

ocinlogon() crea una nueva conexión con una base de datos Oracle 8 e inicia una sesión. El tercer parámetro opcional puede contener el nombre de la instancia a Oracle o el nombre de una entrada en tnsnames.ora a la cual desea conectarse. Si el tercer parámetro opcional no es especificado, PHP usa las variables de entorno ORACLE_SID (instancia de Oracle) o TWO_TASK (tnsnames.ora) para determinar a cuál base de datos conectarse.

ocinlogon() obliga a crear una nueva conexión. Debe usar usada si necesita aislar un conjunto de transacciones. Por defecto, las conexiones son compartidas al nivel de página si usa **ocilogon()** o al nivel de proceso del servidor web si usa **ociplogon()**. Si tiene múltiples conexiones abiertas usando **ocinlogon()**, todos los procesos "commit" y "rollback" se aplican sólo a la conexión especificada.

Si usais Oracle server v.9.2 ó superior, podeis definir el parametro *charset*, el cual será utilizado en la nueva conexión. Si utilizais una version < 9.2, este parámetro será ignorado y la variable de entorno NLS_LANG será usada en su lugar.

Este ejemplo demuestra el modo en que las conexiones son separadas.
Ejemplo 1. Ejemplo de ocinlogon()

```php
<?php
echo "<html><pre>";
$bd = "";

$c1 = ocilogon("scott", "tiger", $bd);
$c2 = ocinlogon("scott", "tiger", $bd);

function crear_tabla($con)
{
  $sentencia = ociparse($con, "create table scott.hallo (test
varchar2(64))");
  ociexecute($sentencia);
  echo $con . " tabla creada\n\n";
}

function abandonar_tabla($con)
{
  $sentencia = ociparse($con, "drop table scott.hallo");
  ociexecute($sentencia);
  echo $con . " tabla abandonada\n\n";
}

function insertar_datos($con)
{
  $sentencia = ociparse($con, "insert into scott.hallo
           values('$con' || ' ' || to_char(sysdate,'DD-MON-YY HH24:MI:SS'))");
  ociexecute($sentencia, OCI_DEFAULT);
  echo $con . "  datos insertados en hallo\n\n";
}

function eliminar_datos($con)
{
  $sentencia = ociparse($con, "delete from scott.hallo");
  ociexecute($sentencia, OCI_DEFAULT);
  echo $con . " datos eliminados de hallo\n\n";
}

function commit($con)
{
  ocicommit($con);
  echo $con . " aplicado\n\n";
}

function rollback($con)
{
  ocirollback($con);
  echo $con . " rollback\n\n";
}

function seleccionar_datos($con)
{
  $sentencia = ociparse($con, "select * from scott.hallo");
  ociexecute($sentencia, OCI_DEFAULT);
  echo $con . "----seleccionando\n\n";
  while (ocifetch($sentencia)) {
    echo $con . " <" . ociresult($sentencia, "TEST") . ">\n\n";
  }
  echo $con . "----listo\n\n";
}

crear_tabla($c1);
insertar_datos($c1);

seleccionar_datos($c1);
seleccionar_datos($c2);

rollback($c1);

seleccionar_datos($c1);
seleccionar_datos($c2);

insertar_datos($c2);
```

Vea también **ocilogon()** y **ociplogon()**.

OCINumCols

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

OCINumCols -- Devuelve el número de columnas resultantes en una sentencia

Descripción

int **OCINumCols** (int stmt)

OCINumCols() devuelve el número de columnas en una sentencia

Ejemplo 1. OCINumCols

```
<?php
    print "<HTML><PRE>\n";
    $conn = OCILogon("scott", "tiger");
    $stmt = OCIParse($conn,"select * from emp");
    OCIExecute($stmt);
    while ( OCIFetch($stmt) ) {
        print "\n";
        $ncols = OCINumCols($stmt);
        for ( $i = 1; $i <= $ncols; $i++ ) {
            $column_name  = OCIColumnName($stmt,$i);
            $column_value = OCIResult($stmt,$i);
            print $column_name . ': ' . $column_value . "\n";
        }
        print "\n";
    }
    OCIFreeStatement($stmt);
    OCILogoff($conn);
    print "</PRE>";
    print "</HTML>\n";
?>
```

OCIParse

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

OCIParse -- Analiza una consulta y devuelve una sentencia

Descripción

int **OCIParse** (int conn, strint query)

OCIParse() analiza la *query* usando *conn*. Devuelve el identificador de la sentencia si la consulta es válida, y falso si no lo es. La *query* puede ser cualquier sentencia SQL válida.

ociplogon

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

ociplogon -- Conectarse con una base de datos Oracle usando una conexión persistente

Descripción

resource **ociplogon** (string nombre_usuario, string contrasenya [, string bd [, string juego_caracteres]])

ociplogon() crea una conexión persistente con una base de datos Oracle 8 e inicia una sesión. El tercer parámetro opcional puede contener el nombre de la instancia local a Oracle o el nombre de la entrada en tnsnames.ora a la cual desea conectarse. Si el tercer parámetro opcional no se especifica, PHP usa las variables de entorno *ORACLE_SID* (instancia de Oracle) o *TWO_TASK* (tnsnames.ora) para determinar a cuál base de datos conectarse.

Si usais Oracle server v.9.2 ó superior, podeis definir el parametro *charset*, el cual será utilizado en la nueva conexión. Si utilizais una version < 9.2, este parámetro será ignorado y la variable de entorno NLS_LANG será usada en su lugar.

Vea también **ocilogon()** y **ocinlogon()**.

OCIResult

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

OCIResult -- Devuelve el valor de una columna en la fila buscada

Descripción

mixed **OCIResult** (int statement, mixed column)

OCIResult() devuelve el valor de la columna *column* de la fila actual (vea **OCIFetch()**).**OCIResult()** devolverá todo como una cadena excepto para los tipo de datos abstractos (ROWIDs, LOBs and FILEs).

OCIRollback

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

OCIRollback -- Restablece todas las transaciones sin confirmar

Descripción

int **OCIRollback** (int connection)

OCIRollback() restablece todas las transacciones sin confirmar para la conexión Oracle *connection*.

OCIRowCount

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

OCIRowCount -- Obtiene el número de filas afectadas

Descripción

int **OCIRowCount** (int statement)

OCIRowCount() devuelve el número de filas afectadas, por ej. en sentencias de actualización. ! Esta función no indicará el número de de filas que devuelve una sentencia SELECT!

Ejemplo 1. OCIRowCount

```
<?php
    print "<HTML><PRE>";
    $conn = OCILogon("scott","tiger");
    $stmt = OCIParse($conn,"create table emp2 as select * from emp");
    OCIExecute($stmt);
    print OCIRowCount($stmt) . " rows inserted.<BR>";
    OCIFreeStatement($stmt);
    $stmt = OCIParse($conn,"delete from emp2");
    OCIExecute($stmt);
    print OCIRowCount($stmt) . " rows deleted.<BR>";
    OCICommit($conn);
    OCIFreeStatement($stmt);
    $stmt = OCIParse($conn,"drop table emp2");
    OCIExecute($stmt);
    OCIFreeStatement($stmt);
    OCILogOff($conn);
    print "</PRE></HTML>";
?>
```

ocisavelob

(PHP 4 , PHP 5)

ocisavelob -- Saves a large object

Description

bool **ocisavelob** (void)

> **Nota:** This function was renamed to **oci_lob_save()** after PHP >= 5.0.0. For downward compatibility **ocisavelob()** can also be used. This is deprecated, however.

ocisavelobfile

(PHP 4 , PHP 5)

ocisavelobfile -- Saves a large object file

Description

bool **ocisavelobfile** (void)

> **Nota:** This function was renamed to **oci_lob_import()** after PHP >= 5.0.0. For downward compatibility **ocisavelobfile()** can also be used. This is deprecated, however.

OCIServerVersion

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

OCIServerVersion -- Devuelve una cadena conteniendo información a cerca de la version del servidor.

Descripción

string **OCIServerVersion** (int conn)

Ejemplo 1. OCIServerVersion

```
<?php
   $conn = OCILogon("scott","tiger");
   print "Server Version: " . OCIServerVersion($conn);
   OCILogOff($conn);
?>
```

lob->setBuffering

(no version information, might be only in CVS)

lob->setBuffering -- Changes current state of buffering for large object

Description

bool **lob->setBuffering** (bool on_off)

lob->setBuffering() sets the buffering for the large object, depending on the value of the *on_off* parameter. Repeated calls to **lob->setBuffering()** with the same flag will return **`TRUE`**. The values for *on_off* are: **`TRUE`** for on and **`FALSE`** for off.

Use of this function may provide performance improvements by buffering small reads and writes of LOBs by reducing the number of network round-trips and LOB versions. **oci_lob_flush()** should be used to flush buffers, when you have finished working with the large object.

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

See also **ocigetbufferinglob()**.

ocisetprefetch

(PHP 3>= 3.0.12, PHP 4 , PHP 5)

ocisetprefetch -- Sets number of rows to be prefetched

Description

bool **ocisetprefetch** (resource stmt, int rows)

Sets the number of top level rows to be prefetched to *rows*. The default value for *rows* is 1 row.

> **Nota:** This function was renamed to **oci_set_prefetch()** after PHP >= 5.0.0. For downward compatibility **ocisetprefetch()** can also be used. This is deprecated, however.

OCIStatementType

(PHP 3>= 3.0.5, PHP 4 , PHP 5)

OCIStatementType -- Devuelve el tipo de una sentencia OCI.

Descripción

string **OCIStatementType** (int stmt)

OCIStatementType() devuelve uno de los siguiente valores:

1. "SELECT"
2. "UPDATE"
3. "DELETE"
4. "INSERT"
5. "CREATE"
6. "DROP"
7. "ALTER"
8. "BEGIN"
9. "DECLARE"
10. "UNKNOWN"

Ejemplo 1. Code examples

```
<?php
    print "<HTML><PRE>";
    $conn = OCILogon("scott","tiger");
    $sql  = "delete from emp where deptno = 10";

    $stmt = OCIParse($conn,$sql);
    if ( OCIStatementType($stmt) == "DELETE" ) {
        die "You are not allowed to delete from this table<BR>";
    }

    OCILogoff($conn);
    print "</PRE></HTML>";
?>
```

ociwritelobtofile

(PHP 4 , PHP 5)

ociwritelobtofile -- Saves a large object file

Description

bool **ociwritelobtofile** ([string filename [, int start [, int length]]])

> **Nota:** This function was renamed to **oci_lob_export()** after PHP >= 5.0.0. For downward compatibility **ociwritelobtofile()** can also be used. This is deprecated, however.

ociwritetemporarylob

(no version information, might be only in CVS)

ociwritetemporarylob -- Writes temporary blob

Description

bool **ociwritetemporarylob** (string var [, int lob_type])

> **Nota:** This function was renamed to **oci_lob_write_temporary()** after PHP >= 5.0.0. For downward compatibility **ociwritetemporarylob()** can also be used. This is deprecated, however.

LXXXVIII. OpenAL Audio Bindings

Introducción

Platform independent audio bindings. Requires the OpenAL library.

Instalación

Esta extension PECL no esta ligada a PHP.

Mas informacion sobre nuevos lanzamientos, descargas ficheros de fuentes, informacion sobre los responsables asi como un 'CHANGELOG', se puede encontrar aqui: http://pecl.php.net/package/openal.

Podeis descargar esta DLL de las extensiones PECL desde la pagina PHP Downloads o desde http://snaps.php.net/.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

This extension defines four resource types: *Open AL(Device)* - Returned by **openal_device_open()**, *Open AL(Context)* - Returned by **openal_context_create()**, *Open AL(Buffer)* - Returned by **openal_buffer_create()**, and *Open AL(Source)* - Returned by **openal_source_create()**.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

`ALC_FREQUENCY` (**integer**)

Context Attribute

`ALC_REFRESH` (**integer**)

Context Attribute

`ALC_SYNC` (**integer**)

Context Attribute

`AL_FREQUENCY` (**integer**)

Buffer Setting

`AL_BITS` (**integer**)

Buffer Setting

`AL_CHANNELS` (**integer**)

Buffer Setting

`AL_SIZE` (**integer**)

Buffer Setting

`AL_BUFFER` (**integer**)

Source/Listener Setting (Integer)

`AL_SOURCE_RELATIVE` (**integer**)

Source/Listener Setting (Integer)

AL_SOURCE_STATE (integer)

Source/Listener Setting (Integer)

AL_PITCH (integer)

Source/Listener Setting (Float)

AL_GAIN (integer)

Source/Listener Setting (Float)

AL_MIN_GAIN (integer)

Source/Listener Setting (Float)

AL_MAX_GAIN (integer)

Source/Listener Setting (Float)

AL_MAX_DISTANCE (integer)

Source/Listener Setting (Float)

AL_ROLLOFF_FACTOR (integer)

Source/Listener Setting (Float)

AL_CONE_OUTER_GAIN (integer)

Source/Listener Setting (Float)

AL_CONE_INNER_ANGLE (integer)

Source/Listener Setting (Float)

AL_CONE_OUTER_ANGLE (integer)

Source/Listener Setting (Float)

AL_REFERENCE_DISTANCE (integer)

Source/Listener Setting (Float)

AL_POSITION (integer)

Source/Listener Setting (Float Vector)

AL_VELOCITY (integer)

Source/Listener Setting (Float Vector)

AL_DIRECTION (integer)

Source/Listener Setting (Float Vector)

AL_ORIENTATION (integer)

Source/Listener Setting (Float Vector)

AL_FORMAT_MONO8 (integer)

PCM Format

AL_FORMAT_MONO16 (integer)

PCM Format

AL_FORMAT_STEREO8 (integer)

PCM Format

AL_FORMAT_STEREO16 (integer)

PCM Format

AL_INITIAL (integer)

Source State

AL_PLAYING (integer)

Source State

AL_PAUSED (integer)

Source State

AL_STOPPED (integer)

Source State

AL_LOOPING (integer)

Source State

AL_TRUE (integer)

Boolean value recognized by OpenAL

AL_FALSE (integer)

Boolean value recognized by OpenAL

Tabla de contenidos

openal_buffer_create

(no version information, might be only in CVS)

openal_buffer_create -- Generate OpenAL buffer

Descripción

resource **openal_buffer_create** (void)

Valores retornados

Returns an Open AL(Buffer) resource on success or **FALSE** on failure.

Ver también

openal_buffer_loadwav
()
openal_buffer_data()

openal_buffer_data

(no version information, might be only in CVS)

openal_buffer_data -- Load a buffer with data

Descripción

bool **openal_buffer_data** (resource buffer, int format, string data, int freq)

Lista de parámetros

buffer

An Open AL(Buffer) resource (previously created by **openal_buffer_create()**).

format

Format of *data*, one of: **AL_FORMAT_MONO8**, **AL_FORMAT_MONO16**, **AL_FORMAT_STEREO8**, y **AL_FORMAT_STEREO16**

data

Block of binary audio data in the *format* and *freq* specified.

frew

Frequency of *data* given in Hz.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ver también

openal_buffer_loadwav()
openal_stream()

openal_buffer_destroy

(no version information, might be only in CVS)

openal_buffer_destroy -- Destroys an OpenAL buffer

Descripción

bool **openal_buffer_destroy** (resource buffer)

Lista de parámetros

buffer

An Open AL(Buffer) resource (previously created by **openal_buffer_create()**).

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ver también

openal_buffer_create()

openal_buffer_get

(no version information, might be only in CVS)

openal_buffer_get -- Retrieve an OpenAL buffer property

Descripción

int **openal_buffer_get** (resource buffer, int property)

Lista de parámetros

buffer

An Open AL(Buffer) resource (previously created by **openal_buffer_create()**).

property

Specific property, one of: **AL_FREQUENCY**, **AL_BITS**, **AL_CHANNELS**, y **AL_SIZE**.

Valores retornados

Returns an integer value appropriate to the *property* requested or **FALSE** on failure.

Ver también

openal_buffer_create()

openal_buffer_loadwav

(no version information, might be only in CVS)

openal_buffer_loadwav -- Load a .wav file into a buffer

Descripción

bool **openal_buffer_loadwav** (resource buffer, string wavfile)

Lista de parámetros

buffer

An Open AL(Buffer) resource (previously created by **openal_buffer_create()**).

wavfile

Path to .WAV file on *local* file system.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ver también

openal_buffer_data()
openal_stream()

openal_context_create

(no version information, might be only in CVS)

openal_context_create -- Create an audio processing context

Descripción

resource **openal_context_create** (resource device)

Lista de parámetros

device

An Open AL(Device) resource (previously created by **openal_device_open()**).

Valores retornados

Returns an Open AL(Context) resource on success or **FALSE** on failure.

Ver también

openal_device_open()
openal_context_destroy()

openal_context_current

(no version information, might be only in CVS)

openal_context_current -- Make the specified context current

Descripción

bool **openal_context_current** (resource context)

Lista de parámetros

context

An Open AL(Context) resource (previously created by **openal_context_create()**).

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ver también

openal_context_create()

openal_context_destroy

(no version information, might be only in CVS)

openal_context_destroy -- Destroys a context

Descripción

bool **openal_context_destroy** (resource context)

Lista de parámetros

context

An Open AL(Context) resource (previously created by **openal_context_create()**).

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ver también

openal_context_create()

openal_context_process

(no version information, might be only in CVS)

openal_context_process -- Process the specified context

Descripción

bool **openal_context_process** (resource context)

Lista de parámetros

context

An Open AL(Context) resource (previously created by **openal_context_create()**).

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ver también

openal_context_create()
openal_context_current()
openal_context_suspend()

openal_context_suspend

(no version information, might be only in CVS)

openal_context_suspend -- Suspend the specified context

Descripción

bool **openal_context_suspend** (resource context)

Lista de parámetros

context

An Open AL(Context) resource (previously created by **openal_context_create()**).

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ver también

openal_context_create()
openal_context_current()
openal_context_process()

openal_device_close

(no version information, might be only in CVS)

openal_device_close -- Close an OpenAL device

Descripción

bool **openal_device_close** (resource device)

Lista de parámetros

device

An Open AL(Device) resource (previously created by **openal_device_open()**) to be closed.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ver también

openal_device_open()

openal_device_open

(no version information, might be only in CVS)

openal_device_open -- Initialize the OpenAL audio layer

Descripción

resource **openal_device_open** ([string device_desc])

Lista de parámetros

device_desc

Open an audio device optionally specified by *device_desc*. If *device_desc* is not specified the first available audio device will be used.

Valores retornados

Returns an Open AL(Device) resource on success or **FALSE** on failure.

Ver también

openal_device_close()

openal_context_create()

openal_listener_get

(no version information, might be only in CVS)

openal_listener_get -- Retrieve a listener property

Descripción

mixed **openal_listener_get** (int property)

Lista de parámetros

property

Property to retrieve, one of: **AL_GAIN** (float), **AL_POSITION** (array(float,float,float)), **AL_VELOCITY** (array(float,float,float)), y **AL_ORIENTATION** (array(float,float,float)).

Valores retornados

Returns a float or array of floats (as appropriate), or **FALSE** on failure.

Ver también

openal_listener_set()

openal_listener_set

(no version information, might be only in CVS)

openal_listener_set -- Set a listener property

Descripción

bool **openal_listener_set** (int property, mixed setting)

Lista de parámetros

property

Property to set, one of: **AL_GAIN** (float), **AL_POSITION** (array(float,float,float)), **AL_VELOCITY** (array(float,float,float)), y **AL_ORIENTATION** (array(float,float,float)).

setting

Value to set, either float, or an array of floats as appropriate.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ver también

openal_listener_get()

openal_source_create

(no version information, might be only in CVS)

openal_source_create -- Generate a source resource

Descripción

resource **openal_source_create** (void)

Valores retornados

Returns an Open AL(Source) resource on success or **FALSE** on failure.

Ver también

openal_source_set()
openal_source_play()
openal_source_destroy()

openal_source_destroy

(no version information, might be only in CVS)

openal_source_destroy -- Destroy a source resource

Descripción

resource **openal_source_destroy** (resource source)

Lista de parámetros

source

An Open AL(Source) resource (previously created by **openal_source_create()**).

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ver también

openal_source_create()

openal_source_get

(no version information, might be only in CVS)

openal_source_get -- Retrieve an OpenAL source property

Descripción

mixed **openal_source_get** (resource source, int property)

Lista de parámetros

source

An Open AL(Source) resource (previously created by **openal_source_create()**).

property

Property to get, one of: **AL_SOURCE_RELATIVE** (int), **AL_SOURCE_STATE** (int), **AL_PITCH** (float), **AL_GAIN** (float), **AL_MIN_GAIN** (float), **AL_MAX_GAIN** (float), **AL_MAX_DISTANCE** (float), **AL_ROLLOFF_FACTOR** (float), **AL_CONE_OUTER_GAIN** (float), **AL_CONE_INNER_ANGLE** (float), **AL_CONE_OUTER_ANGLE** (float), **AL_REFERENCE_DISTANCE** (float), **AL_POSITION** (array(float,float,float)), **AL_VELOCITY** (array(float,float,float)), **AL_DIRECTION** (array(float,float,float)).

Valores retornados

Returns the type associated with the property being retrieved or **FALSE** on failure.

Ver también

openal_source_create()
openal_source_set()
openal_source_play()

openal_source_pause

(no version information, might be only in CVS)

openal_source_pause -- Pause the source

Descripción

bool **openal_source_pause** (resource source)

Lista de parámetros

source

An Open AL(Source) resource (previously created by **openal_source_create()**).

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ver también

openal_source_stop()
openal_source_play()
openal_source_rewind()

openal_source_play

(no version information, might be only in CVS)

openal_source_play -- Start playing the source

Descripción

bool **openal_source_play** (resource source)

Lista de parámetros

source

An Open AL(Source) resource (previously created by **openal_source_create()**).

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ver también

openal_source_stop()
openal_source_pause()
openal_source_rewind()

openal_source_rewind

(no version information, might be only in CVS)

openal_source_rewind -- Rewind the source

Descripción

bool **openal_source_rewind** (resource source)

Lista de parámetros

source

An Open AL(Source) resource (previously created by **openal_source_create()**).

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ver también

openal_source_stop()
openal_source_pause()
openal_source_play()

openal_source_set

(no version information, might be only in CVS)

openal_source_set -- Set source property

Descripción

bool **openal_source_set** (resource source, int property, mixed setting)

Lista de parámetros

source

An Open AL(Source) resource (previously created by **openal_source_create()**).

property

Property to set, one of: **AL_BUFFER** (OpenAL(Source)), **AL_LOOPING** (bool), **AL_SOURCE_RELATIVE** (int), **AL_SOURCE_STATE** (int), **AL_PITCH** (float), **AL_GAIN** (float), **AL_MIN_GAIN** (float), **AL_MAX_GAIN** (float), **AL_MAX_DISTANCE** (float), **AL_ROLLOFF_FACTOR** (float), **AL_CONE_OUTER_GAIN** (float), **AL_CONE_INNER_ANGLE** (float), **AL_CONE_OUTER_ANGLE** (float), **AL_REFERENCE_DISTANCE** (float), **AL_POSITION** (array(float,float,float)), **AL_VELOCITY** (array(float,float,float)), **AL_DIRECTION** (array(float,float,float)).

setting

Value to assign to specified *property*. Refer to the description of *property* for a description of

the value(s) expected.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ver también

openal_source_create()
openal_source_get()
openal_source_play()

openal_source_stop

(no version information, might be only in CVS)

openal_source_stop -- Stop playing the source

Descripción

bool **openal_source_stop** (resource source)

Lista de parámetros

source

An Open AL(Source) resource (previously created by **openal_source_create()**).

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ver también

openal_source_play()
openal_source_pause()
openal_source_rewind()

openal_stream

(no version information, might be only in CVS)

openal_stream -- Begin streaming on a source

Descripción

resource **openal_stream** (resource source, int format, int rate)

Lista de parámetros

source

An Open AL(Source) resource (previously created by **openal_source_create()**).

format

Format of *data*, one of: **AL_FORMAT_MONO8**, **AL_FORMAT_MONO16**, **AL_FORMAT_STEREO8**, y **AL_FORMAT_STEREO16**

rate

Frequency of data to stream given in Hz.

Valores retornados

Returns a stream resource on success, or **FALSE** on failure.

Ver también

openal_source_create()
fwrite()

LXXXIX. OpenSSL Functions

Introducción

This module uses the functions of OpenSSL for generation and verification of signatures and for sealing (encrypting) and opening (decrypting) data. OpenSSL offers many features that this module currently doesn't support. Some of these may be added in the future.

Requirimientos

In order to use the OpenSSL functions you need to install the OpenSSL package. PHP between versions 4.0.5 and 4.3.1 will work with OpenSSL >= 0.9.5. Other versions (PHP <=4.0.4pl1 and >= 4.3.2) require OpenSSL >= 0.9.6.

Aviso
You are strongly encouraged to use the most recent OpenSSL version, otherwise your web server could be vulnerable to attack.

Instalación

To use PHP's OpenSSL support you must also compile PHP *--with-openssl[=DIR]*.

Note to Win32 Users: In order to enable this module on a Windows environment, you

must copy `libeay32.dll` from the DLL folder of the PHP/Win32 binary package to the SYSTEM32 folder of your windows machine. (Ex: `C:\WINNT\SYSTEM32` or `C:\WINDOWS\SYSTEM32`)

Additionally, if you are planning to use the key generation and certificate signing functions, you will need to install a valid `openssl.cnf` on your system. As of PHP 4.3.0, we include a sample configuration file in the `openssl` folder of our win32 binary distribution. If you are using PHP 4.2.0 or later and are missing the file, you can obtain it from the OpenSSL home page or by downloading the PHP 4.3.0 release and using the configuration file from there.

Note to Win32 Users: PHP will search for the `openssl.cnf` using the following logic:

- the *OPENSSL_CONF* environmental variable, if set, will be used as the path (including filename) of the configuration file.
- the *SSLEAY_CONF* environmental variable, if set, will be used as the path (including filename) of the configuration file.
- The file `openssl.cnf` will be assumed to be found in the default certificate area, as configured at the time that the openssl DLL was compiled. This is usually means that the default filename is `c:\usr\local\ssl\openssl.cnf`.

In your installation, you need to decide whether to install the configuration file at `c:\usr\local\ssl\openssl.cnf` or whether to install it someplace else and use environmental variables (possibly on a per-virtual-host basis) to locate the configuration file. Note that it is possible to override the default path from the script using the *configargs* of the functions that require a configuration file.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

Key/Certificate parameters

Quite a few of the openssl functions require a key or a certificate parameter. PHP 4.0.5 and earlier have to use a key or certificate **resource** returned by one of the openssl_get_xxx functions. Later versions may use one of the following methods:

- Certificates
  - An X.509 resource returned from **openssl_x509_read()**
  - A string having the format `file://path/to/cert.pem`; the named file must

contain a PEM encoded certificate

- A string containing the content of a certificate, PEM encoded

- Public/Private Keys
  - A key resource returned from **openssl_get_publickey()** or **openssl_get_privatekey()**
  - For public keys only: an X.509 resource
  - A string having the format `file://path/to/file.pem` - the named file must contain a PEM encoded certificate/private key (it may contain both)
  - A string containing the content of a certificate/key, PEM encoded
  - For private keys, you may also use the syntax *array($key, $passphrase)* where $key represents a key specified using the file:// or textual content notation above, and $passphrase represents a string containing the passphrase for that private key

---

Certificate Verification

When calling a function that will verify a signature/certificate, the *cainfo* parameter is an array containing file and directory names that specify the locations of trusted CA files. If a directory is specified, then it must be a correctly formed hashed directory as the **openssl** command would use.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

Purpose checking flags

`X509_PURPOSE_SSL_CLIENT` (**integer**)

`X509_PURPOSE_SSL_SERVER` (**integer**)

`X509_PURPOSE_NS_SSL_SERVER` (**integer**)

`X509_PURPOSE_SMIME_SIGN` (**integer**)

`X509_PURPOSE_SMIME_ENCRYPT` (**integer**)

`X509_PURPOSE_CRL_SIGN` (**integer**)

`X509_PURPOSE_ANY` (**integer**)

Padding flags

OPENSSL_PKCS1_PADDING (integer)

OPENSSL_SSLV23_PADDING (integer)

OPENSSL_NO_PADDING (integer)

OPENSSL_PKCS1_OAEP_PADDING (integer)

Key types

OPENSSL_KEYTYPE_RSA (integer)

OPENSSL_KEYTYPE_DSA (integer)

OPENSSL_KEYTYPE_DH (integer)

PKCS7 Flags/Constants

The S/MIME functions make use of flags which are specified using a bitfield which can include one or more of the following values:

Tabla 1. PKCS7 CONSTANTS

Constant	Description
PKCS7_TEXT	Adds text/plain content type headers to encrypted/signed message. If decrypting or verifying, it strips those headers from the output - if the decrypted or verified message is not of MIME type text/plain then an error will occur.
PKCS7_BINARY	Normally the input message is converted to "canonical" format which is effectively using CR and LF as end of line: as required by the S/MIME specification. When this options is present, no translation occurs. This is useful when handling binary data which may not be in MIME format.
PKCS7_NOINTERN	When verifying a message, certificates (if any) included in the message are normally searched for the signing certificate. With this option only the certificates specified in the *extracerts* parameter of **openssl_pkcs7_verify()** are used. The supplied certificates can still be used as untrusted CAs however.
PKCS7_NOVERIFY	Do not verify the signers certificate of a signed message.
PKCS7_NOCHAIN	Do not chain verification of signers certificates: that is don't use the certificates in the signed message as untrusted CAs.
PKCS7_NOCERTS	When signing a message the signer's certificate is normally included - with this option it is excluded. This will reduce the size of the signed message but the verifier must have a copy of the signers certificate available locally (passed using the *extracerts* to **openssl_pkcs7_verify()** for example).

Constant	Description
PKCS7_NOATTR	Normally when a message is signed, a set of attributes are included which include the signing time and the supported symmetric algorithms. With this option they are not included.
PKCS7_DETACHED	When signing a message, use cleartext signing with the MIME type multipart/signed. This is the default if you do not specify any *flags* to **openssl_pkcs7_sign()**. If you turn this option off, the message will be signed using opaque signing, which is more resistant to translation by mail relays but cannot be read by mail agents that do not support S/MIME.
PKCS7_NOSIGS	Don't try and verify the signatures on a message

Nota: These constants were added in 4.0.6.

Tabla de contenidos

openssl_csr_export_to_file

(PHP 4 >= 4.2.0, PHP 5)

openssl_csr_export_to_file -- Exports a CSR to a file

Description

bool **openssl_csr_export_to_file** (resource csr, string outfilename [, bool notext])

openssl_csr_export_to_file() takes the Certificate Signing Request represented by *csr* and saves it as ascii-armoured text into the file named by *outfilename*.

The optional parameter *notext* affects the verbosity of the output; if it is **FALSE** then additional human-readable information is included in the output. The default value of *notext* is **TRUE**.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

See also **openssl_csr_export()**, **openssl_csr_new()** and **openssl_csr_sign()**.

openssl_csr_export

(PHP 4 >= 4.2.0, PHP 5)

openssl_csr_export -- Exports a CSR as a string

Description

bool **openssl_csr_export** (resource csr, string &out [, bool notext])

openssl_csr_export() takes the Certificate Signing Request represented by *csr* and stores it as ascii-armoured text into *out*, which is passed by reference.

The optional parameter *notext* affects the verbosity of the output; if it is **FALSE** then additional human-readable information is included in the output. The default value of *notext* is **TRUE**.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

See also **openssl_csr_export_to_file()**, **openssl_csr_new()** and **openssl_csr_sign()**.

openssl_csr_new

(PHP 4 >= 4.2.0, PHP 5)

openssl_csr_new -- Generates a CSR

Description

bool **openssl_csr_new** (array dn, resource &privkey [, array configargs [, array extraattribs]])

openssl_csr_new() generates a new CSR (Certificate Signing Request) based on the information provided by *dn*, which represents the Distinguished Name to be used in the certificate.

privkey should be set to a private key that was previously generated by **openssl_pkey_new()** (or otherwise obtained from the other openssl_pkey family of functions). The corresponding public portion of the key will be used to sign the CSR.

extraattribs is used to specify additional configuration options for the CSR. Both *dn* and *extraattribs* are associative arrays whose keys are converted to OIDs and applied to the relevant part of the request.

> **Nota:** You need to have a valid `openssl.cnf` installed for this function to operate correctly. See the notes under the installation section for more information.

By default, the information in your system *openssl.conf* is used to initialize the request; you can specify a configuration file section by setting the *config_section_section* key of *configargs*. You can also specify an alternative openssl configuration file by setting the value of the *config* key to the path of the file you want to use. The following keys, if present in *configargs* behave as their ñalents in the *openssl.conf*, as listed in the table below.

Tabla 1. Configuration overrides

***configargs* key**	**type**	***openssl.conf* ñalent**	**description**
digest_alg	**string**	default_md	Selects which digest method to use
x509_extensions	**string**	x509_extensions	Selects which extensions should be used when creating an x509 certificate
req_extensions	**string**	req_extensions	Selects which extensions should be used when creating a CSR
private_key_bits	**string**	default_bits	Specifies how many bits should be used to generate a private key
private_key_type	**integer**	none	Specifies the type of private key to create. This can be one of **OPENSSL_KEYTYPE_DSA**, **OPENSSL_KEYTYPE_DH** or **OPENSSL_KEYTYPE_RSA**. The default value is **OPENSSL_KEYTYPE_RSA** which is currently the only supported key type.
encrypt_key	**boolean**	encrypt_key	Should an exported key (with passphrase) be encrypted?

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplo 1. openssl_csr_new() example - creating a self-signed-certificate

```php
<?php
// Fill in data for the distinguished name to be used in the cert
// You must change the values of these keys to match your name and
// company, or more precisely, the name and company of the person/site
// that you are generating the certificate for.
// For SSL certificates, the commonName is usually the domain name of
// that will be using the certificate, but for S/MIME certificates,
// the commonName will be the name of the individual who will use the
// certificate.
$dn = array(
    "countryName" => "UK",
    "stateOrProvinceName" => "Somerset",
    "localityName" => "Glastonbury",
    "organizationName" => "The Brain Room Limited",
    "organizationalUnitName" => "PHP Documentation Team",
    "commonName" => "Wez Furlong",
    "emailAddress" => "wez@example.com"
);

// Generate a new private (and public) key pair
$privkey = openssl_pkey_new();

// Generate a certificate signing request
$csr = openssl_csr_new($dn, $privkey);

// You will usually want to create a self-signed certificate at this
// point until your CA fulfills your request.
// This creates a self-signed cert that is valid for 365 days
$sscert = openssl_csr_sign($csr, null, $privkey, 365);

// Now you will want to preserve your private key, CSR and self-signed
// cert so that they can be installed into your web server, mail server
// or mail client (depending on the intended use of the certificate).
// This example shows how to get those things into variables, but you
// can also store them directly into files.
// Typically, you will send the CSR on to your CA who will then issue
// you with the "real" certificate.
openssl_csr_export($csr, $csrout) and var_dump($csrout);
openssl_x509_export($sscert, $certout) and var_dump($certout);
openssl_pkey_export($privkey, $pkeyout, "mypassword") and var_dump($pkeyout);

// Show any errors that occurred here
while (($e = openssl_error_string()) !== false) {
    echo $e . "\n";
}
?>
```

openssl_csr_sign

(PHP 4 >= 4.2.0, PHP 5)

openssl_csr_sign -- Sign a CSR with another certificate (or itself) and generate a certificate

Description

resource **openssl_csr_sign** (mixed csr, mixed cacert, mixed priv_key, int days [, array configargs [, int serial]])

openssl_csr_sign() generates an x509 certificate resource from the *csr* previously generated by **openssl_csr_new()**, but it can also be the path to a PEM encoded CSR when specified as `file://path/to/csr` or an exported string generated by **openssl_csr_export()**. The generated certificate will be signed by *cacert*. If *cacert* is **NULL**, the generated certificate will be a self-signed certificate. *priv_key* is the private key that corresponds to *cacert*. *days* specifies the length of time

for which the generated certificate will be valid, in days. You can finetune the CSR signing by *configargs*. See **openssl_csr_new()** for more information about *configargs*. Since PHP 4.3.3 you can specify the serial number of issued certificate by *serial*. In earlier versions, it was always 0.

Returns an x509 certificate resource on success, **FALSE** on failure.

> **Nota:** You need to have a valid `openssl.cnf` installed for this function to operate correctly. See the notes under the installation section for more information.

Ejemplo 1. openssl_csr_sign() example - signing a CSR (how to implement your own CA)

```
<?php
// Let's assume that this script is set to receive a CSR that has
// been pasted into a textarea from another page
$csrdata = $_POST["CSR"];

// We will sign the request using our own "certificate authority"
// certificate.  You can use any certificate to sign another, but
// the process is worthless unless the signing certificate is trusted
// by the software/users that will deal with the newly signed certificate

// We need our CA cert and its private key
$cacert = "file://path/to/ca.crt";
$privkey = array("file://path/to/ca.key", "your_ca_key_passphrase");

$userscert = openssl_csr_sign($csrdata, $cacert, $privkey, 365);

// Now display the generated certificate so that the user can
// copy and paste it into their local configuration (such as a file
// to hold the certificate for their SSL server)
openssl_x509_export($usercert, $certout);
echo $certout;

// Show any errors that occurred here
while (($e = openssl_error_string()) !== false) {
    echo $e . "\n";
}
?>
```

openssl_error_string

(PHP 4 >= 4.0.6, PHP 5)

openssl_error_string -- Return openSSL error message

Description

mixed **openssl_error_string** (void)

Returns an error message string, or **FALSE** if there are no more error messages to return.

openssl_error_string() returns the last error from the openSSL library. Error messages are stacked, so this function should be called multiple times to collect all of the information.

Ejemplo 1. openssl_error_string() example

```
<?php
// lets assume you just called an openssl function that failed
while ($msg = openssl_error_string())
    echo $msg . "<br />\n";
?>
```

openssl_free_key

(PHP 4 >= 4.0.4, PHP 5)

openssl_free_key -- Free key resource

Description

void **openssl_free_key** (resource key_identifier)

openssl_free_key() frees the key associated with the specified *key_identifier* from memory.

openssl_get_privatekey

(PHP 4 >= 4.0.4, PHP 5)

openssl_get_privatekey -- Get a private key

Description

resource **openssl_get_privatekey** (mixed key [, string passphrase])

This is an alias for **openssl_pkey_get_private()**.

openssl_get_publickey

(PHP 4 >= 4.0.4, PHP 5)

openssl_get_publickey -- Extract public key from certificate and prepare it for use

Description

resource **openssl_get_publickey** (mixed certificate)

This is an alias for **openssl_pkey_get_public()**.

openssl_open

(PHP 4 >= 4.0.4, PHP 5)

openssl_open -- Open sealed data

Description

bool **openssl_open** (string sealed_data, string &open_data, string env_key, mixed priv_key_id)

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo. If successful the opened data is returned in *open_data*.

openssl_open() opens (decrypts) *sealed_data* using the private key associated with the key identifier *priv_key_id* and the envelope key *env_key*, and fills *open_data* with the decrypted data. The envelope key is generated when the data are sealed and can only be used by one specific private key. See **openssl_seal()** for more information.

Ejemplo 1. openssl_open() example

```
<?php
// $sealed and $env_key are assumed to contain the sealed data
// and our envelope key, both given to us by the sealer.

// fetch private key from file and ready it
$fp = fopen("/src/openssl-0.9.6/demos/sign/key.pem", "r");
$priv_key = fread($fp, 8192);
fclose($fp);
$pkeyid = openssl_get_privatekey($priv_key);

// decrypt the data and store it in $open
if (openssl_open($sealed, $open, $env_key, $pkeyid)) {
    echo "here is the opened data: ", $open;
} else {
    echo "failed to open data";
}

// free the private key from memory
openssl_free_key($pkeyid);
?>
```

See also **openssl_seal()**.

openssl_pkcs7_decrypt

(PHP 4 >= 4.0.6, PHP 5)

openssl_pkcs7_decrypt -- Decrypts an S/MIME encrypted message

Description

bool **openssl_pkcs7_decrypt** (string infilename, string outfilename, mixed recipcert [, mixed recipkey])

Decrypts the S/MIME encrypted message contained in the file specified by *infilename* using the certificate and its associated private key specified by *recipcert* and *recipkey*.

The decrypted message is output to the file specified by *outfilename*

Ejemplo 1. openssl_pkcs7_decrypt() example

```
<?php
// $cert and $key are assumed to contain your personal certificate and private
// key pair, and that you are the recipient of an S/MIME message
$infilename = "encrypted.msg";  // this file holds your encrypted message
$outfilename = "decrypted.msg"; // make sure you can write to this file

if (openssl_pkcs7_decrypt($infilename, $outfilename, $cert, $key)) {
    echo "decrypted!";
} else {
    echo "failed to decrypt!";
}
?>
```

openssl_pkcs7_encrypt

(PHP 4 >= 4.0.6, PHP 5)

openssl_pkcs7_encrypt -- Encrypt an S/MIME message

Description

bool **openssl_pkcs7_encrypt** (string infile, string outfile, mixed recipcerts, array headers [, int flags [, int cipherid]])

openssl_pkcs7_encrypt() takes the contents of the file named *infile* and encrypts them using an RC2 40-bit cipher so that they can only be read by the intended recipients specified by *recipcerts*, which is either a lone X.509 certificate, or an array of X.509 certificates. *headers* is an array of headers that will be prepended to the data after it has been encrypted. *flags* can be used to specify options that affect the encoding process - see PKCS7 constants. *headers* can be either an associative array keyed by header name, or an indexed array, where each element contains a single header line. Cipher can be selected with *cipherid* since PHP 5.

Ejemplo 1. openssl_pkcs7_encrypt() example

```
<?php
// the message you want to encrypt and send to your secret agent
// in the field, known as nighthawk.  You have his certificate
// in the file nighthawk.pem
$data = <<<EOD
Nighthawk,

Top secret, for your eyes only!

The enemy is closing in! Meet me at the cafe at 8.30am
to collect your forged passport!

HQ
EOD;

// load key
$key = file_get_contents("nighthawk.pem");

// save message to file
$fp = fopen("msg.txt", "w");
fwrite($fp, $data);
fclose($fp);

// encrypt it
if (openssl_pkcs7_encrypt("msg.txt", "enc.txt", $key,
    array("To" => "nighthawk@example.com", // keyed syntax
          "From: HQ <hq@example.com>", // indexed syntax
          "Subject" => "Eyes only"))) {
    // message encrypted - send it!
    exec(ini_get("sendmail_path") . " < enc.txt");
}
?>
```

openssl_pkcs7_sign

(PHP 4 >= 4.0.6, PHP 5)

openssl_pkcs7_sign -- Sign an S/MIME message

Description

bool **openssl_pkcs7_sign** (string infilename, string outfilename, mixed signcert, mixed privkey, array headers [, int flags [, string extracerts]])

openssl_pkcs7_sign() takes the contents of the file named *infilename* and signs them using the certificate and its matching private key specified by *signcert* and *privkey* parameters.

headers is an array of headers that will be prepended to the data after it has been signed (see **openssl_pkcs7_encrypt()** for more information about the format of this parameter.

flags can be used to alter the output - see PKCS7 constants - if not specified, it defaults to PKCS7_DETACHED.

extracerts specifies the name of a file containing a bunch of extra certificates to include in the signature which can for example be used to help the recipient to verify the certificate that you used.

Ejemplo 1. openssl_pkcs7_sign() example

```
<?php
// the message you want to sign so that recipient can be sure it was you that
// sent it
$data = <<<EOD

You have my authorization to spend $10,000 on dinner expenses.

The CEO
EOD;
// save message to file
$fp = fopen("msg.txt", "w");
fwrite($fp, $data);
fclose($fp);
// encrypt it
if (openssl_pkcs7_sign("msg.txt", "signed.txt", "mycert.pem",
    array("file://mycert.pem", "mypassphrase"),
    array("To" => "joes@example.com", // keyed syntax
          "From: HQ <ceo@example.com>", // indexed syntax
          "Subject" => "Eyes only")
    )) {
    // message signed - send it!
    exec(ini_get("sendmail_path") . " < signed.txt");
}
?>
```

openssl_pkcs7_verify

(PHP 4 >= 4.0.6, PHP 5)

openssl_pkcs7_verify -- Verifies the signature of an S/MIME signed message

Description

bool **openssl_pkcs7_verify** (string filename, int flags [, string outfilename [, array cainfo [, string extracerts]]])

openssl_pkcs7_verify() reads the S/MIME message contained in the filename specified by *filename* and examines the digital signature. It returns **TRUE** if the signature is verified, **FALSE** if it is not correct (the message has been tampered with, or the signing certificate is invalid), or -1 on

error.

flags can be used to affect how the signature is verified - see PKCS7 constants for more information.

If the *outfilename* is specified, it should be a string holding the name of a file into which the certificates of the persons that signed the messages will be stored in PEM format.

If the *cainfo* is specified, it should hold information about the trusted CA certificates to use in the verification process - see certificate verification for more information about this parameter.

If the *extracerts* is specified, it is the filename of a file containing a bunch of certificates to use as untrusted CAs.

openssl_pkey_export_to_file

(PHP 4 >= 4.2.0, PHP 5)

openssl_pkey_export_to_file -- Gets an exportable representation of a key into a file

Description

bool **openssl_pkey_export_to_file** (mixed key, string outfilename [, string passphrase [, array configargs]])

openssl_pkey_export_to_file() saves an ascii-armoured (PEM encoded) rendition of *key* into the file named by *outfilename*. The key can be optionally protected by a *passphrase*. *configargs* can be used to fine-tune the export process by specifying and/or overriding options for the openssl configuration file. See **openssl_csr_new()** for more information about *configargs*. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

> **Nota:** You need to have a valid `openssl.cnf` installed for this function to operate correctly. See the notes under the installation section for more information.

openssl_pkey_export

(PHP 4 >= 4.2.0, PHP 5)

openssl_pkey_export -- Gets an exportable representation of a key into a string

Description

bool **openssl_pkey_export** (mixed key, string &out [, string passphrase [, array configargs]])

openssl_pkey_export() exports *key* as a PEM encoded string and stores it into *out* (which is passed by reference). The key is optionally protected by *passphrase*. *configargs* can be used to fine-tune the export process by specifying and/or overriding options for the openssl configuration file. See **openssl_csr_new()** for more information about *configargs*. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

> **Nota:** You need to have a valid `openssl.cnf` installed for this function to operate

correctly. See the notes under the installation section for more information.

openssl_pkey_get_private

(PHP 4 >= 4.2.0, PHP 5)

openssl_pkey_get_private -- Get a private key

Description

resource **openssl_pkey_get_private** (mixed key [, string passphrase])

Returns a positive key resource identifier on success, or **FALSE** on error.

openssl_get_privatekey() parses *key* and prepares it for use by other functions. *key* can be one of the following:

1. a string having the format `file://path/to/file.pem`. The named file must contain a PEM encoded certificate/private key (it may contain both).

2. A PEM formatted private key.

The optional parameter *passphrase* must be used if the specified key is encrypted (protected by a passphrase).

openssl_pkey_get_public

(PHP 4 >= 4.2.0, PHP 5)

openssl_pkey_get_public -- Extract public key from certificate and prepare it for use

Description

resource **openssl_pkey_get_public** (mixed certificate)

Returns a positive key resource identifier on success, or **FALSE** on error.

openssl_get_publickey() extracts the public key from *certificate* and prepares it for use by other functions. *certificate* can be one of the following:

1. an X.509 certificate resource

2. a string having the format `file://path/to/file.pem`. The named file must contain a PEM encoded certificate/private key (it may contain both).

3. A PEM formatted private key.

openssl_pkey_new

(PHP 4 >= 4.2.0, PHP 5)

openssl_pkey_new -- Generates a new private key

Description

resource **openssl_pkey_new** ([array configargs])

openssl_pkey_new() generates a new private and public key pair. The public component of the key can be obtained using **openssl_pkey_get_public()**. You can finetune the key generation (such as specifying the number of bits) using *configargs*. See **openssl_csr_new()** for more information about *configargs*.

> **Nota:** You need to have a valid `openssl.cnf` installed for this function to operate correctly. See the notes under the installation section for more information.

openssl_private_decrypt

(PHP 4 >= 4.0.6, PHP 5)

openssl_private_decrypt -- Decrypts data with private key

Description

bool **openssl_private_decrypt** (string data, string &decrypted, mixed key [, int padding])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

openssl_private_decrypt() decrypts *data* that was previous encrypted via **openssl_public_encrypt()** and stores the result into *decrypted*. *key* must be the private key corresponding that was used to encrypt the data. *padding* defaults to **`OPENSSL_PKCS1_PADDING`**, but can also be one of **`OPENSSL_SSLV23_PADDING`**, **`OPENSSL_PKCS1_OAEP_PADDING`**, **`OPENSSL_NO_PADDING`**.

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

You can use this function e.g. to decrypt data which were supposed only to you.

See also **openssl_public_encrypt()** and **openssl_public_decrypt()**.

openssl_private_encrypt

(PHP 4 >= 4.0.6, PHP 5)

openssl_private_encrypt -- Encrypts data with private key

Description

bool **openssl_private_encrypt** (string data, string &crypted, mixed key [, int padding])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

openssl_private_encrypt() encrypts *data* with private *key* and stores the result into *crypted*. Encrypted data can be decrypted via **openssl_public_decrypt()**. *padding* defaults to **OPENSSL_PKCS1_PADDING**, but can also be **OPENSSL_NO_PADDING**.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

This function can be used e.g. to sign data (or its hash) to prove that it is not written by someone else.

See also **openssl_public_decrypt()** and **openssl_public_encrypt()**.

openssl_public_decrypt

(PHP 4 >= 4.0.6, PHP 5)

openssl_public_decrypt -- Decrypts data with public key

Description

bool **openssl_public_decrypt** (string data, string &decrypted, mixed key [, int padding])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

openssl_public_decrypt() decrypts *data* that was previous encrypted via **openssl_private_encrypt()** and stores the result into *decrypted*. *key* must be the public key corresponding that was used to encrypt the data. *padding* defaults to **OPENSSL_PKCS1_PADDING**, but can also be **OPENSSL_NO_PADDING**.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

You can use this function e.g. to check if the message was written by the owner of the private key.

See also **openssl_private_encrypt()** and **openssl_private_decrypt()**.

openssl_public_encrypt

(PHP 4 >= 4.0.6, PHP 5)

openssl_public_encrypt -- Encrypts data with public key

Description

bool **openssl_public_encrypt** (string data, string &crypted, mixed key [, int padding])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

openssl_public_encrypt() encrypts *data* with public *key* and stores the result into *crypted*. Encrypted data can be decrypted via **openssl_private_decrypt()**. *padding* defaults to **OPENSSL_PKCS1_PADDING**, but can also be one of **OPENSSL_SSLV23_PADDING**, **OPENSSL_PKCS1_OAEP_PADDING**, **OPENSSL_NO_PADDING**.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

This function can be used e.g. to encrypt message which can be then read only by owner of the private key. It can be also used to store secure data in database.

See also **openssl_private_decrypt()** and **openssl_private_encrypt()**.

openssl_seal

(PHP 4 >= 4.0.4, PHP 5)

openssl_seal -- Seal (encrypt) data

Description

int **openssl_seal** (string data, string &sealed_data, array &env_keys, array pub_key_ids)

Returns the length of the sealed data on success, or **FALSE** on error. If successful the sealed data is returned in *sealed_data*, and the envelope keys in *env_keys*.

openssl_seal() seals (encrypts) *data* by using RC4 with a randomly generated secret key. The key is encrypted with each of the public keys associated with the identifiers in *pub_key_ids* and each encrypted key is returned in *env_keys*. This means that one can send sealed data to multiple recipients (provided one has obtained their public keys). Each recipient must receive both the sealed data and the envelope key that was encrypted with the recipient's public key.

Ejemplo 1. openssl_seal() example

```
<?php
// $data is assumed to contain the data to be sealed

// fetch public keys for our recipients, and ready them
$fp = fopen("/src/openssl-0.9.6/demos/maurice/cert.pem", "r");
$cert = fread($fp, 8192);
fclose($fp);
$pk1 = openssl_get_publickey($cert);
// Repeat for second recipient
$fp = fopen("/src/openssl-0.9.6/demos/sign/cert.pem", "r");
$cert = fread($fp, 8192);
fclose($fp);
$pk2 = openssl_get_publickey($cert);

// seal message, only owners of $pk1 and $pk2 can decrypt $sealed with keys
// $ekeys[0] and $ekeys[1] respectively.
openssl_seal($data, $sealed, $ekeys, array($pk1, $pk2));

// free the keys from memory
openssl_free_key($pk1);
openssl_free_key($pk2);
?>
```

See also **openssl_open()**.

openssl_sign

(PHP 4 >= 4.0.4, PHP 5)

openssl_sign -- Generate signature

Description

bool **openssl_sign** (string data, string &signature, mixed priv_key_id [, int signature_alg])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo. If successful the signature is returned in *signature*.

openssl_sign() computes a signature for the specified *data* by using SHA1 for hashing followed by encryption using the private key associated with *priv_key_id*. Note that the data itself is not encrypted.

Ejemplo 1. openssl_sign() example

```
<?php
// $data is assumed to contain the data to be signed

// fetch private key from file and ready it
$fp = fopen("/src/openssl-0.9.6/demos/sign/key.pem", "r");
$priv_key = fread($fp, 8192);
fclose($fp);
$pkeyid = openssl_get_privatekey($priv_key);

// compute signature
openssl_sign($data, $signature, $pkeyid);

// free the key from memory
openssl_free_key($pkeyid);
?>
```

See also **openssl_verify()**.

openssl_verify

(PHP 4 >= 4.0.4, PHP 5)

openssl_verify -- Verify signature

Description

int **openssl_verify** (string data, string signature, mixed pub_key_id)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Returns 1 if the signature is correct, 0 if it is incorrect, and -1 on error.

openssl_verify() verifies that the *signature* is correct for the specified *data* using the public key associated with *pub_key_id*. This must be the public key corresponding to the private key used for signing.

Ejemplo 1. openssl_verify() example

```
<?php
// $data and $signature are assumed to contain the data and the signature

// fetch public key from certificate and ready it
$fp = fopen("/src/openssl-0.9.6/demos/sign/cert.pem", "r");
$cert = fread($fp, 8192);
fclose($fp);
$pubkeyid = openssl_get_publickey($cert);

// state whether signature is okay or not
$ok = openssl_verify($data, $signature, $pubkeyid);
if ($ok == 1) {
    echo "good";
} elseif ($ok == 0) {
    echo "bad";
} else {
    echo "ugly, error checking signature";
}
// free the key from memory
openssl_free_key($pubkeyid);
?>
```

See also **openssl_sign()**.

openssl_x509_check_private_key

(PHP 4 >= 4.2.0, PHP 5)

openssl_x509_check_private_key -- Checks if a private key corresponds to a certificate

Description

bool **openssl_x509_check_private_key** (mixed cert, mixed key)

openssl_x509_check_private_key() returns **TRUE** if *key* is the private key that corresponds to *cert*, or **FALSE** otherwise.

openssl_x509_checkpurpose

(PHP 4 >= 4.0.6, PHP 5)

openssl_x509_checkpurpose -- Verifies if a certificate can be used for a particular purpose

Description

bool **openssl_x509_checkpurpose** (mixed x509cert, int purpose [, array cainfo [, string untrustedfile]])

Returns **TRUE** if the certificate can be used for the intended purpose, **FALSE** if it cannot, or -1 on error.

openssl_x509_checkpurpose() examines the certificate specified by *x509cert* to see if it can be used for the purpose specified by *purpose*.

cainfo should be an array of trusted CA files/dirs as described in Certificate Verification. It defaults to an empty array.

untrustedfile, if specified, is the name of a PEM encoded file holding certificates that can be used to help verify the certificate, although no trust in placed in the certificates that come from that file.

Tabla 1. openssl_x509_checkpurpose() purposes

Constant	Description
X509_PURPOSE_SSL_CLIENT	Can the certificate be used for the client side of an SSL connection?
X509_PURPOSE_SSL_SERVER	Can the certificate be used for the server side of an SSL connection?
X509_PURPOSE_NS_SSL_SERVER	Can the cert be used for Netscape SSL server?
X509_PURPOSE_SMIME_SIGN	Can the cert be used to sign S/MIME email?
X509_PURPOSE_SMIME_ENCRYPT	Can the cert be used to encrypt S/MIME email?
X509_PURPOSE_CRL_SIGN	Can the cert be used to sign a certificate revocation list (CRL)?
X509_PURPOSE_ANY	Can the cert be used for Any/All purposes?

These options are not bitfields - you may specify one only!

openssl_x509_export_to_file

(PHP 4 >= 4.2.0, PHP 5)

openssl_x509_export_to_file -- Exports a certificate to file

Description

bool **openssl_x509_export_to_file** (mixed x509, string outfilename [, bool notext])

openssl_x509_export_to_file() stores *x509* into a file named by *outfilename* in a PEM encoded format.

The optional parameter *notext* affects the verbosity of the output; if it is **FALSE** then additional human-readable information is included in the output. The default value of *notext* is **TRUE**.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

openssl_x509_export

(PHP 4 >= 4.2.0, PHP 5)

openssl_x509_export -- Exports a certificate as a string

Description

bool **openssl_x509_export** (mixed x509, string &output [, bool notext])

openssl_x509_export() stores *x509* into a string named by *output* in a PEM encoded format.

The optional parameter *notext* affects the verbosity of the output; if it is **FALSE** then additional human-readable information is included in the output. The default value of *notext* is **TRUE**.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

openssl_x509_free

(PHP 4 >= 4.0.6, PHP 5)

openssl_x509_free -- Free certificate resource

Description

void **openssl_x509_free** (resource x509cert)

openssl_x509_free() frees the certificate associated with the specified *x509cert* resource from memory.

openssl_x509_parse

(PHP 4 >= 4.0.6, PHP 5)

openssl_x509_parse -- Parse an X509 certificate and return the information as an array

Description

array **openssl_x509_parse** (mixed x509cert [, bool shortnames])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

openssl_x509_parse() returns information about the supplied *x509cert*, including fields such as subject name, issuer name, purposes, valid from and valid to dates etc. *shortnames* controls how the data is indexed in the array - if *shortnames* is **TRUE** (the default) then fields will be indexed with the short name form, otherwise, the long name form will be used - e.g.: CN is the shortname form of commonName.

The structure of the returned data is (deliberately) not yet documented, as it is still subject to change.

openssl_x509_read

(PHP 4 >= 4.0.6, PHP 5)

openssl_x509_read -- Parse an X.509 certificate and return a resource identifier for it

Description

resource **openssl_x509_read** (mixed x509certdata)

openssl_x509_read() parses the certificate supplied by *x509certdata* and returns a resource identifier for it.

XC. Funciones Oracle

Tabla de contenidos

Ora_Bind

(PHP 3, PHP 4 , PHP 5)

Ora_Bind -- Vincula una variable PHP a un parámetro Oracle

Descripción

int **ora_bind** (int cursor, string nombre de variable PHP, string nombre de parámetro SQL, int longitud [, int tipo])

Devuelve verdadero si el vínculo se realiza con éxito, y sino devuelve falso. Los detalles de los errores pueden examinarse usando la funciones **ora_error()** y **ora_errorcode()**.

Esta función liga la variable PHP nombrada con el parámetro SQL. El parámtro SQL debe estar en la forma ":name". Con el parámetro optativo tipo, se define si el parámetro SQL se trata de un parámetro de entrada/salida (0 y por defecto), entrada (1) o salida (2). Como en PHP 3.0.1, se puede

usar las constantes ORA_BIND_INOUT, ORA_BIND_IN y ORA_BIND_OUT en lugar de los números.

ora_bind debe ser llamada después de **ora_parse()** y antes de **ora_exec()**. Los valores de entrada pueden pasarse por asignación a las variables PHP vinculadas, despues de la llamada a **ora_exec()** dichas variables contendrán los valores de salida, si éstos estuvieran disponibles.

```
<?php
ora_parse($curs, "declare tmp INTEGER; begin tmp := :in; :out := tmp; :x := 7.77; end;"
ora_bind($curs, "result", ":x", $len, 2);
ora_bind($curs, "input", ":in", 5, 1);
ora_bind($curs, "output", ":out", 5, 2);
$input = 765;
ora_exec($curs);
echo "Result: $result<BR>Out: $output<BR>In: $input";
?>
```

Ora_Close

(PHP 3, PHP 4 , PHP 5)

Ora_Close -- Cierra un cursor Oracle

Descripción

int **ora_close** (int cursor)

Devuelve verdadero si el cierre fué exitoso, o falso de lo contrario. Los detalles de los errores se recuperan usando las funciones **ora_error()** y **ora_errorcode()**.

Esta función cierra un cursor de datos abierto con **ora_open()**.

Ora_ColumnName

(PHP 3, PHP 4 , PHP 5)

Ora_ColumnName -- toma el nombre de una columna de resultados Oracle

Descripción

string **Ora_ColumnName** (int cursor, int column)

Devuelve el nomre de un campo/columna *column* en el cursor *cursor*. el nombre devuelto estará en letras mayúsculas.

ora_columnsize

(PHP 3, PHP 4 , PHP 5)

ora_columnsize -- Returns the size of an Oracle result column

Description

int **ora_columnsize** (resource cursor, int column)

Returns the size of the Oracle column *column* on the cursor *cursor*. Column 0 is the first column.

Ora_ColumnType

(PHP 3, PHP 4 , PHP 5)

Ora_ColumnType -- toma el tipo de una columna de resultados Oracle

Descripción

string **Ora_ColumnType** (int cursor, int column)

Devuelve el nombre del tipo de datos del campo o columna *column* en el cursor *cursor*. Se devolverá un tipo de datos, de entre los siguientes:

"VARCHAR2"
"VARCHAR"
"CHAR"
"NUMBER"
"LONG"
"LONG RAW"
"ROWID"
"DATE"
"CURSOR"

Ora_Commit

(PHP 3, PHP 4 , PHP 5)

Ora_Commit -- realiza una transacción Oracle

Descripción

int **ora_commit** (int conn)

Devuelve verdadero si es exitosa, de lo cantrario devuelve falso. Puede verse los detalles del error usando las funciones **ora_error()** y **ora_errorcode()**. Esta función realiza una transacción Oracle. Se define como transacción cualquier cambio en una conexión dada, desde la última tarea/retroceso en la ejecucion (rollback), anulación de la ejecución automática de tareas (autocommit), o cuando se ha establecido la conexión.

Ora_CommitOff

(PHP 3, PHP 4 , PHP 5)

Ora_CommitOff -- deshabilita el modo automatico de ejecucion de tareas (autocommit)

Descripción

int **ora_commitoff** (int conn)

Devuelve verdadero si se ejecuta con éxito, sino devuelve falso. Los pormenores del error en cuestion, pueden revisarse invocando las funciones **ora_error()** y **ora_errorcode()**.

Esta función deshabilita la ejecucion automatica luego de cada instancia **ora_exec()**.

Ora_CommitOn

(PHP 3, PHP 4 , PHP 5)

Ora_CommitOn -- Habilita la ejecucion automática de tareas (autocommit)

Descripción

int **ora_commiton** (int conn)

Esta función habilita la ejecucion automatica luego de cada instancia **ora_exec()** en la conexión dada.

Devuelve verdadero si se ejecuta con éxito, sino devuelve falso. Los pormenores del error en cuestion, pueden revisarse invocando las funciones **ora_error()** y **ora_errorcode()**.

ora_do

(PHP 3, PHP 4 , PHP 5)

ora_do -- Parse, Exec, Fetch

Description

resource **ora_do** (resource conn, string query)

ora_do() is quick combination of **ora_parse()**, **ora_exec()** and **ora_fetch()**. It will parse and execute a statement, then fetch the first result row.

This function returns a cursor index or **FALSE** on failure. Details about the error can be retrieved using the **ora_error()** and **ora_errorcode()** functions.

See also **ora_parse()**,**ora_exec()**, and **ora_fetch()**.

Ora_Error

(PHP 3, PHP 4 , PHP 5)

Ora_Error -- toma los mensajes de error de Oracle

Descripción

string **Ora_Error** (int cursor_or_connection)

Devuelve los mensajes de error en la forma *XXX-NNNNN* donde *XXX* es la procedencia del error y *NNNNN* es la identificación del mensaje de error.

> **Nota:** El soporte para las identificaciones de conexión fue agregado en la versión 3.0.4.

En las versiones UNIX de Oracle, pueden encontrarse detalles acerca de un mensaje de error como este: `$ oerr ora 00001 00001, 00000, "unique constraint (%s.%s) violated" // *Cause: An update or insert statement attempted to insert a duplicate key // For Trusted ORACLE configured in DBMS MAC mode, you may see // this message if a duplicate entry exists at a different level. // *Action: Either remove the unique restriction or do not insert the key`

Ora_ErrorCode

(PHP 3, PHP 4 , PHP 5)

Ora_ErrorCode -- captura el código de error Oracle

Descripción

int **Ora_ErrorCode** (int cursor_or_connection)

Devuelve el codigo numérico de error de la última declaración ejecutada en el cursor o conexion espacificada.

> **Nota:** El soporte para las identificaciones de conexión fue agregado en la versión 3.0.4.

Ora_Exec

(PHP 3, PHP 4 , PHP 5)

Ora_Exec -- ejecuta las declaraciones interpretadas en un cursor Oracle

Descripción

int **ora_exec** (int cursor)

Devuelve verdadero ante la ejecución exitosa, de lo contrario, devuelve falso. Los detalles del error pueden verse invocando las funciones **ora_error()** y **ora_errorcode()**.

ora_fetch_into

(PHP 3, PHP 4 , PHP 5)

ora_fetch_into -- Fetch a row into the specified result array

Description

int **ora_fetch_into** (resource cursor, array &result [, int flags])

Fetches a row of data into an array. The *flags* has two flag values: if the **ORA_FETCHINTO_NULLS** flag is set, columns with **NULL** values are set in the array; and if the **ORA_FETCHINTO_ASSOC** flag is set, an associative array is created.

Returns the number of columns fetched.

Ejemplo 1. ora_fetch_into()

```
<?php
$results = array();
ora_fetch_into($cursor, $results);
echo $results[0];
echo $results[1];
$results = array();
ora_fetch_into($cursor, $results, ORA_FETCHINTO_NULLS|ORA_FETCHINTO_ASSOC);
echo $results['MyColumn'];
?>
```

See also **ora_parse()**,**ora_exec()**, **ora_fetch()**, and **ora_do()**.

Ora_Fetch

(PHP 3, PHP 4 , PHP 5)

Ora_Fetch -- extrae una fila de datos a partir de un cursor

Descripción

int **ora_fetch** (int cursor)

Devuelve verdadero (se extrajo una fila) o falso (no hay mas filas, o ha ocurrido un error). Si ocurre un error, los detalles del mismo pueden verse invocando las funciones **ora_error()** y **ora_errorcode()**. Si no hubo errores, **ora_errorcode()** devolverá 0. Recupera una hilera de datos partiendo de un cursor especificado.

Ora_GetColumn

(PHP 3, PHP 4 , PHP 5)

Ora_GetColumn -- toma datos de la fila extraída

Descripción

mixed **ora_getcolumn** (int cursor, mixed column)

Devuelve la columna de datos. Si hay un error, se devuelve falso y **ora_errorcode()** devulve un valor distinto de cero. Note, de igual manera, que la busqueda de un resultado Falso en esta función, puede resultar verdadera, aún en casos en que no ocurran errores:(resultado NULO, cadenas vacias,

valor 0 o cadenas "0"). Extrae los datos para una columna o resultado de función.

Ora_Logoff

(PHP 3, PHP 4 , PHP 5)

Ora_Logoff -- cierra una conexión Oracle

Descripción

int **ora_logoff** (int connection)

Devuelve verdadero si es exitosa, o falso si hay errores. Los detalles del error pueden verse invocando las funciones **ora_error()** y **ora_errorcode()**. Cierra la sesión de trabajo del usuario, y lo desconecta del servidor.

Ora_Logon

(PHP 3, PHP 4 , PHP 5)

Ora_Logon -- Abre una conexión Oracle

Descripción

int **ora_logon** (string usuario, string contraseña)

Establece una conexión entre PHP y una base de datos Oracle, con los datos de nombre de usuario y contraseña especificados.

Las conexiones pueden llevarse adelante usando SQL*Net indicando el nombre TNS al *usuario* de este modo:

```
$conn = Ora_Logon("usuario@TNSNAME", "contraseña");
```

Si hubiesen datos con caracteres no-ASCII, habría que asegurarse de que esté presente la variable de entorno *NLS_LANG* en el sistema. Para los modulos de servidor, deberían definirse en el entorno del servidor antes de iniciarlo.

Devuelve el índice de la conexión si aquella tuvo éxito, de lo cantrario devuelve falso. Los detalles del error pueden verse invocando las funciones **ora_error()** y **ora_errorcode()**.

ora_numcols

(PHP 3, PHP 4 , PHP 5)

ora_numcols -- Returns the number of columns

Description

int **ora_numcols** (resource cursor)

ora_numcols() returns the number of columns in a result. Only returns meaningful values after an parse/exec/fetch sequence.

See also **ora_parse()**,**ora_exec()**, **ora_fetch()**, and **ora_do()**.

ora_numrows

(PHP 3, PHP 4 , PHP 5)

ora_numrows -- Returns the number of rows

Description

int **ora_numrows** (resource cursor)

ora_numrows() returns the number of rows in a result.

Ora_Open

(PHP 3, PHP 4 , PHP 5)

Ora_Open -- abre un cursor Oracle

Descripción

int **ora_open** (int connection)

Abre un cursor asociado con la conexión.

Devuelve el índice del cursor o Falso si hay un error. Los detalles del error pueden verse invocando las funciones **ora_error()** y **ora_errorcode()**.

Ora_Parse

(PHP 3, PHP 4 , PHP 5)

Ora_Parse -- interpreta una declaración SQL

Descripción

int **ora_parse** (int cursor_ind, string sql_statement, int defer)

Esta función interpreta una declaración SQL o un bloque PL/SQL y los asocia con el cursor dado. Devuelve 0 si se ejecuta con exito, o -1 ante un error.

ora_plogon

(PHP 3, PHP 4 , PHP 5)

ora_plogon -- Open a persistent Oracle connection

Description

resource **ora_plogon** (string user, string password)

Establishes a persistent connection between PHP and an Oracle database with the username *user* and password *password*.

See also **ora_logon()**.

Ora_Rollback

(PHP 3, PHP 4 , PHP 5)

Ora_Rollback -- retrocede en la lista de transacciones (hace un roll back)

Descripción

int **ora_rollback** (int connection)

Deshace una transaccion Oracle. (Ver **ora_commit()** para la definición de transacción.)

Devuelve verdadero si tiene éxito, o falso si hay un error. Los detalles del error pueden verse invocando las funciones **ora_error()** y **ora_errorcode()**.

XCI. Funciones de Control de Salida

Introducción

Las funciones de Control de Salida le permiten controlar cuándo es enviada la salida desde el script. Esto puede resultar útil en muchas situaciones diferentes, especialmente si necesita enviar cabeceras al navegador después de que su script ha comenzado a enviar datos. Las funciones de Control de Salida no afectan las cabeceras enviadas usando **header()** o **setcookie()**, sólo funciones como **echo ()** y los datos entre bloques de código PHP.

Requirimientos

No se necesitan bibliotecas externas para construir esta extensión

---

Instalación

No se necesita ninguna instalación para usar estas funciones, son parte del núcleo de PHP.

Configuración en tiempo de ejecución

El comportamiento de estas funciones está afectado por los valores definidos en `php.ini`.

Tabla 1. Opciones de configuración del Control de Salida

Nombre	Predeterminado	Modificable
output_buffering	"0"	PHP_INI_PERDIR| PHP_INI_SYSTEM
output_handler	NULL	PHP_INI_PERDIR| PHP_INI_SYSTEM
implicit_flush	"0"	PHP_INI_PERDIR| PHP_INI_SYSTEM

Para más detalles sobre las constantes PHP_INI_* y su definición, vea **ini_set()**.

A continuación se presenta una corta explicación de las directivas de configuración.

output_buffering **boolean**/**integer**

Puede habilitar el uso de búferes de salida para todos los archivos, definiendo esta directiva como 'On'. Si desea limitar el tamaño del búfer a cierta cantidad - puede usar un número máximo de bytes, en lugar de 'On', como valor para esta directiva (p.ej., output_buffering=4096).

output_handler **string**

Puede redireccionar toda la salida de sus scripts a una función. Por ejemplo, si establece el valor de output_handler a **mb_output_handler()**, la codificación de caracteres será convertida de forma transparente a la codificación especificada. Al definir cualquier gestor de salida, el control mediante búferes se activa automáticamente.

Nota: Usted no puede usar **mb_output_handler()** con **ob_iconv_handler()** al tiempo, y no puede usar **ob_gzhandler()** y zlib.output_compression al tiempo.

implicit_flush **boolean**

FALSE por defecto. Cambiar este valor a **TRUE** le indica a PHP que debe decirle a la capa de salida que se vacíe automáticamente después de cada bloque de salida. Esto es ñalente a llamar la función de PHP **flush()** después de todas y cada una de las llamadas a **print()** o **echo()**, y cada bloque *HTML*.

Cuando use PHP bajo un entorno web, el habilitar esta opción tiene unas implicaciones serias en el rendimiento, y por lo general se recomienda su uso únicamente para propósitos de depuración. Este valor es igual a **TRUE** por defecto cuando se opera bajo la *SAPI CLI*.

Vea también **ob_implicit_flush()**.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Esta extensión no tiene ninguna constante definida.

Ejemplos

Ejemplo 1. Ejemplo de Control de Salida

```
<?php

ob_start();
echo "Hola\n";

setcookie("nombre_cookie", "datos_cookie");

ob_end_flush();

?>
```

En el anterior ejemplo, la salida de **echo()** sería almacenada en el búfer de salida hasta que **ob_end_flush()** sea llamada. Entre tanto, la llamada a **setcookie()** almacena satisfactoriamente una cookie sin causar un error. (Normalmente, usted no puede enviar cabeceras al navegador después de que se han enviado datos.)

> **Nota:** Cuando se actualice desde PHP 4.1 (y 4.2) hacia 4.3, debe asegurarse de que *implict_flush* tenga el valor *OFF* en su `php.ini` debido a un fallo en versiones antiguas, de otra forma, cualquier salida procesada por **ob_start()** no será ocultada en la salida final.

Ver también

Vea también **header()** y **setcookie()**.

Tabla de contenidos

flush

(PHP 3, PHP 4 , PHP 5)

flush -- Flush the output buffer

Description

void **flush** (void)

Flushes the output buffers of PHP and whatever backend PHP is using (CGI, a web server, etc.) This effectively tries to push all the output so far to the user's browser.

ob_clean

(PHP 4 >= 4.2.0, PHP 5)

ob_clean -- Limpiar (eliminar) el búfer de salida

Descripción

void **ob_clean** (void)

Esta función descarta los contenidos del búfer de salida.

Esta función no destruye el búfer de salida, como lo hace **ob_end_clean()**.

Vea también **ob_flush()**, **ob_end_flush()** y **ob_end_clean()**.

ob_end_clean

(PHP 4 , PHP 5)

ob_end_clean -- Clean (erase) the output buffer and turn off output buffering

Description

void **ob_end_clean** (void)

This function discards the contents of the output buffer and turns off output buffering.

See also **ob_start()** and **ob_end_flush()**.

ob_end_flush

(PHP 4 , PHP 5)

ob_end_flush -- Flush (send) the output buffer and turn off output buffering

Description

void **ob_end_flush** (void)

This function will send the contents of the output buffer (if any) and turn output buffering off. If you want to further process the buffer's contents you have to call **ob_get_contents()** before **ob_end_flush()** as the buffer contents are discarded after **ob_get_contents()** is called.

See also **ob_start()**, **ob_get_contents()**, and **ob_end_clean()**.

ob_flush

(PHP 4 >= 4.2.0, PHP 5)

ob_flush -- Vaciar (enviar) el búfer de salida

Descripción

void **ob_flush** (void)

Esta función enviará los contenidos del búfer de salida (si los hay). Si desea continuar procesando los contenidos del búfer, tiene que llamar **ob_get_contents()** antes de **ob_flush()**, ya que los contenidos del búfer son descartados luego de que la función **ob_flush()** es llamada.

Esta función no destruye el búfer de salida, como lo hace **ob_end_flush()**.

Vea también **ob_get_contents()**, **ob_clean()**, **ob_end_flush()** y **ob_end_clean()**.

ob_get_clean

(PHP 4 >= 4.3.0, PHP 5)

ob_get_clean -- Obtener los contenidos del búfer actual y eliminar el búfer de salida actual

Descripción

string **ob_get_clean** (void)

Esta función devolverá los contenidos del búfer de salida y finalizará el control de salida mediante búferes. Si el control de salida con búferes no está activo, entonces se devuelve **FALSE**.
ob_get_clean() básicamente ejecuta tanto **ob_get_contents()** como **ob_end_clean()**.

Ejemplo 1. Un ejemplo simple de ob_get_clean()

```php
<?php

ob_start();

echo "Hola Mundo";

$salida = ob_get_clean();
$salida = strtolower($salida);

var_dump($salida);
?>
```

Nuestro ejemplo producirá la salida:

```
string(10) "hola mundo"
```

Vea también **ob_start()** y **ob_get_contents()**.

ob_get_contents

(PHP 4 , PHP 5)

ob_get_contents -- Devolver el contenido del búfer de salida

Descripción

string **ob_get_contents** (void)

Esta función devolverá los contenidos del búfer de salida, o **FALSE**, si no está activo el uso del búfer de salida.

Vea también **ob_start()** y **ob_get_length()**.

ob_get_flush

(PHP 4 >= 4.3.0, PHP 5)

ob_get_flush -- Flush the output buffer, return it as a string and turn off output buffering

Description

string **ob_get_flush** (void)

ob_get_flush() flushs the output buffer, return it as a string and turns off output buffering. **ob_get_flush()** returns **FALSE** if no buffering is active.

> **Nota:** This function is similar to **ob_end_flush()**, except that this function returns the buffer as a string.

Ejemplo 1. ob_get_flush() example

```
<?php
//using output_buffering=On
print_r(ob_list_handlers());

//save buffer in a file
$buffer = ob_get_flush();
file_put_contents('buffer.txt', $buffer);

print_r(ob_list_handlers());
?>
```

The above example will output:

```
Array
(
    [0] => default output handler
)
Array
(
)
```

See also **ob_end_clean()**, **ob_end_flush()** and **ob_list_handlers()**.

ob_get_length

(PHP 4 >= 4.0.2, PHP 5)

ob_get_length -- Return the length of the output buffer

Description

string **ob_get_length** (void)

This will return the length of the contents in the output buffer or **FALSE**, if output buffering isn't active.

See also **ob_start()** and **ob_get_contents()**.

ob_get_level

(PHP 4 >= 4.2.0, PHP 5)

ob_get_level -- Devolver el nivel de anidamiento del mecanismo de búferes de salida

Descripción

int **ob_get_level** (void)

Esta función devolverá el nivel de gestores de búferes de control anidados o cero si el uso de búferes de salida se encuentra desactivado.

Vea también **ob_start()** y **ob_get_contents()**.

ob_get_status

(PHP 4 >= 4.2.0, PHP 5)

ob_get_status -- Obtener el status de los búferes de salida

Descripción

array **ob_get_status** ([bool status_completo])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Esta función devolverá el status actual de los búferes de salida. Devuelve una matriz que contiene el estátus de búfer, o **FALSE** en caso de fallo.

Vea también **ob_get_level()**.

ob_gzhandler

(PHP 4 >= 4.0.4, PHP 5)

ob_gzhandler -- Llamada de retorno de ob_start para comprimir mediante gzip el búfer de salida

Descripción

string **ob_gzhandler** (string bufer, int modo)

El propósito de **ob_gzhandler()** es el de ser usado como una llamada de retorno para **ob_start()**, facilitando el envío de datos codificados mediante gz hacia navegadores web que soportan la gestión de páginas web comprimidas. Antes de que **ob_gzhandler()** envíe realmente los datos, determina qué tipo de codificación de contenido acepta el navegador ("gzip", "deflate" o ninguno) y devolverá su salida acordemente. Todos los navegadores son soportados, ya que es tarea del navegador el enviar la cabecera apropiada indicando que acepta páginas web comprimidas.

> **Nota:** *modo* fue añadido en PHP 4.0.5.

Ejemplo 1. Ejemplo de ob_gzhandler()

```
<?php

ob_start("ob_gzhandler");

?>
<html>
<body>
<p>Esta debe ser una p&aacute;gina web comprimida.</p>
</html>
<body>
```

> **Nota:** No puede usar **ob_gzhandler()** y zlib.output_compression al mismo tiempo. También note que el uso de zlib.output_compression se prefiere sobre el uso de **ob_gzhandler()**.

Vea también **ob_start()** y **ob_end_flush()**.

ob_implicit_flush

(PHP 4 , PHP 5)

ob_implicit_flush -- Turn implicit flush on/off

Description

void **ob_implicit_flush** ([int flag])

ob_implicit_flush() will turn implicit flushing on or off (if no *flag* is given, it defaults to on). Implicit flushing will result in a flush operation after every output call, so that explicit calls to **flush ()** will no longer be needed.

Turning implicit flushing on will disable output buffering, the output buffers current output will be sent as if **ob_end_flush()** had been called.

See also **flush()**, **ob_start()**, and **ob_end_flush()**.

ob_list_handlers

(PHP 4 >= 4.3.0, PHP 5)

ob_list_handlers -- List all output handlers in use

Description

array **ob_list_handlers** (void)

This will return an array with the output handlers in use (if any). If output_buffering is enabled, **ob_list_handlers()** will return "default output handler".

Ejemplo 1. ob_list_handlers() example

```
<?php
//using output_buffering=On
print_r(ob_list_handlers());
ob_end_flush();

ob_start("ob_gzhandler");
print_r(ob_list_handlers());
ob_end_flush();
?>
```

The above example will output:

```
Array
(
    [0] => default output handler
)
Array
(
    [0] => ob_gzhandler
)
```

See also **ob_end_clean()**, **ob_end_flush()**, **ob_get_flush()**, **ob_start()**.

ob_start

(PHP 4 , PHP 5)

ob_start -- Turn on output buffering

Description

void **ob_start** (void)

This function will turn output buffering on. While output buffering is active no output is sent from the script, instead the output is stored in an internal buffer.

The contents of this internal buffer may be copied into a string variable using **ob_get_contents()**. To output what is stored in the internal buffer, use **ob_end_flush()**. Alternatively, **ob_end_clean()** will silently discard the buffer contents.

See also **ob_get_contents()**, **ob_end_flush()**, **ob_end_clean()**, and **ob_implicit_flush()**

output_add_rewrite_var

(PHP 4 >= 4.3.0, PHP 5)

output_add_rewrite_var -- Add URL rewriter values

Description

bool **output_add_rewrite_var** (string name, string value)

This function rewrite the URLs and forms with the given variable.

> **Nota:** This function buffers the output.

Ejemplo 1. output_add_rewrite_var() example

```
<?php
output_add_rewrite_var('var', 'value');

// a link
echo '<a href="file.php">link</a>';

// a form
echo '<form action="script.php" method="post">
<input type="text" name="var2" />
</form>';

print_r(ob_list_handlers());
?>
```

The above example will output:

```
<a href="file.php?var=value">link</a>

<form action="script.php" method="post">
<input type="hidden" name="var" value="value" />
<input type="text" name="var2" />
</form>

Array
(
    [0] => URL-Rewriter
)
```

See also **output_reset_rewrite_vars()**, **ob_flush()** and **ob_list_handlers()**.

output_reset_rewrite_vars

(PHP 4 >= 4.3.0, PHP 5)

output_reset_rewrite_vars -- Reset URL rewriter values

Description

bool **output_reset_rewrite_vars** (void)

This function resets the URL rewriter and undo the changes made by **output_add_rewrite_var()** and/or by **session_start()** that are still in the buffer.

Ejemplo 1. output_reset_rewrite_vars() example

```
<?php
session_start();
output_add_rewrite_var('var', 'value');

echo '<a href="file.php">link</a>';
ob_flush();

output_reset_rewrite_vars();
echo '<a href="file.php">link</a>';
?>
```

The above example will output:

```
<a href="file.php?PHPSESSID=xxx&var=value">link</a>
<a href="file.php">link</a>
```

See also **output_add_rewrite_var()**, **ob_flush()**, **ob_list_handlers()** and **session_start()**.

XCII. Object property and method call overloading

Introducción

The purpose of this extension is to allow overloading of object property access and method calls. Only one function is defined in this extension, **overload()** which takes the name of the class that should have this functionality enabled. The class named has to define appropriate methods if it wants to have this functionality: *__get()*, *__set()* and *__call()* respectively for getting/setting a property, or calling a method. This way overloading can be selective. Inside these handler functions the overloading is disabled so you can access object properties normally.

Aviso
Esta extensión es *EXPERIMENTAL*. Esto significa que el comportamiento de esta extensión, los nombre de sus funciones y en definitiva TODO lo documentado sobre esta extensión, puede cambiar en una futura versión de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
This extension is not a part of PHP 5. PHP 5 supports *__get()*, *__set()* and *__call()* natively. See the Overloading in PHP 5 page for more information.

Requirimientos

No se necesitan bibliotecas externas para construir esta extensión

Instalación

In order to use these functions, you must compile PHP with the *--enable-overload* option. Starting with PHP 4.3.0 this extension is enabled by default. You can disable overload support with *--disable--overload*.

La versión para Windows de *PHP* tiene soporte nativo para esta extensión. No se necesita cargar ninguna extensión adicional para usar estas funciones.

> **Nota:** Builtin support for overload is available with PHP 4.3.0.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Esta extensión no tiene ninguna constante definida.

Ejemplos

Some simple examples on using the **overload()** function:
Ejemplo 1. Overloading a PHP class

```php
<?php

class OO {
    var $a = 111;
    var $elem = array('b' => 9, 'c' => 42);

    // Callback method for getting a property
    function __get($prop_name, &$prop_value)
    {
        if (isset($this->elem[$prop_name])) {
            $prop_value = $this->elem[$prop_name];
            return true;
        } else {
            return false;
        }
    }

    // Callback method for setting a property
    function __set($prop_name, $prop_value)
    {
        $this->elem[$prop_name] = $prop_value;
        return true;
    }
}

// Here we overload the OO object
overload('OO');

$o = new OO;
echo "\$o->a: $o->a\n"; // print: $o->a: 111
echo "\$o->b: $o->b\n"; // print: $o->b: 9
echo "\$o->c: $o->c\n"; // print: $o->c: 42
echo "\$o->d: $o->d\n"; // print: $o->d:

// add a new item to the $elem array in OO
$o->x = 56;

// instantiate stdclass (it is built-in in PHP 4)
// $val is not overloaded!
$val = new stdclass;
$val->prop = 555;

// Set "a" to be an array with the $val object in it
// But __set() will put this in the $elem array
$o->a = array($val);
var_dump($o->a[0]->prop);

?>
```

Tabla de contenidos

overload

(PHP 4 >= 4.2.0)

overload -- Enable property and method call overloading for a class

Description

void **overload** ([string class_name])

The **overload()** function will enable property and method call overloading for a class identified by *class_name*. See an example in the introductory section of this part.

XCIII. Ovrimos SQL functions

Ovrimos SQL Server, is a client/server, transactional RDBMS combined with Web capabilities and fast transactions.

Ovrimos SQL Server is available at www.ovrimos.com. To enable ovrimos support in PHP just compile php with the '--with-ovrimos' parameter to configure script. You'll need to install the sqlcli library available in the Ovrimos SQL Server distribution.

Ejemplo 1. Connect to Ovrimos SQL Server and select from a system table

```
<?php
$conn = ovrimos_connect ("server.domain.com", "8001", "admin", "password");
if ($conn != 0) {
    echo ("Connection ok!");
    $res = ovrimos_exec ($conn, "select table_id, table_name from sys.tables");
    if ($res != 0) {
        echo "Statement ok!";
        ovrimos_result_all ($res);
        ovrimos_free_result ($res);
    }
    ovrimos_close($conn);
}
?>
```

This will just connect to SQL Server.

Tabla de contenidos

ovrimos_close

(PHP 4 >= 4.0.3, PHP 5)

ovrimos_close -- Closes the connection to ovrimos

Description

void **ovrimos_close** (int connection)

ovrimos_close() is used to close the specified connection.

ovrimos_close() closes a connection to Ovrimos. This has the effect of rolling back uncommitted transactions.

ovrimos_commit

(PHP 4 >= 4.0.3, PHP 5)

ovrimos_commit -- Commits the transaction

Description

int **ovrimos_commit** (int connection_id)

ovrimos_commit() is used to commit the transaction.

ovrimos_commit() commits the transaction.

ovrimos_connect

(PHP 4 >= 4.0.3, PHP 5)

ovrimos_connect -- Connect to the specified database

Description

int **ovrimos_connect** (string host, string db, string user, string password)

ovrimos_connect() is used to connect to the specified database.

ovrimos_connect() returns a connection id (greater than 0) or 0 for failure. The meaning of 'host' and 'port' are those used everywhere in Ovrimos APIs. 'Host' is a host name or IP address and 'db' is either a database name, or a string containing the port number.

Ejemplo 1. ovrimos_connect() Example

```
<?php
$conn = ovrimos_connect ("server.domain.com", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res=ovrimos_exec ($conn, "select table_id, table_name from sys.tables");
    if ($res != 0) {
        echo "Statement ok!";
        ovrimos_result_all ($res);
        ovrimos_free_result ($res);
    }
    ovrimos_close ($conn);
}
?>
```

The above example will connect to the database and print out the specified table.

ovrimos_cursor

(PHP 4 >= 4.0.3, PHP 5)

ovrimos_cursor -- Returns the name of the cursor

Description

int **ovrimos_cursor** (int result_id)

ovrimos_cursor() is used to get the name of the cursor.

ovrimos_cursor() returns the name of the cursor. Useful when wishing to perform positioned updates or deletes.

ovrimos_exec

(PHP 4 >= 4.0.3, PHP 5)

ovrimos_exec -- Executes an SQL statement

Description

int **ovrimos_exec** (int connection_id, string query)

ovrimos_exec() is used to execute an SQL statement.

ovrimos_exec() executes an SQL statement (query or update) and returns a result_id or **FALSE**. Evidently, the SQL statement should not contain parameters.

ovrimos_execute

(PHP 4 >= 4.0.3, PHP 5)

ovrimos_execute -- Executes a prepared SQL statement

Description

int **ovrimos_execute** (int result_id [, array parameters_array])

ovrimos_execute() is used to execute an SQL statement.

ovrimos_execute() executes a prepared statement. Returns **TRUE** or **FALSE**. If the prepared statement contained parameters (question marks in the statement), the correct number of parameters should be passed in an array. Notice that I don't follow the PHP convention of placing just the name of the optional parameter inside square brackets. I couldn't bring myself on liking it.

ovrimos_fetch_into

(PHP 4 >= 4.0.3, PHP 5)

ovrimos_fetch_into -- Fetches a row from the result set

Description

int **ovrimos_fetch_into** (int result_id, array result_array [, string how [, int rownumber]])

ovrimos_fetch_into() is used to fetch a row from the result set.

ovrimos_fetch_into() fetches a row from the result set into 'result_array', which should be passed by reference. Which row is fetched is determined by the two last parameters. 'how' is one of 'Next' (default), 'Prev', 'First', 'Last', 'Absolute', corresponding to forward direction from current position, backward direction from current position, forward direction from the start, backward direction from the end and absolute position from the start (essentially ñalent to 'first' but needs 'rownumber'). Case is not significant. 'Rownumber' is optional except for absolute positioning. Returns **`TRUE`** or **`FALSE`**.

Ejemplo 1. A fetch into example

```
<?php
$conn=ovrimos_connect ("neptune", "8001", "admin", "password");
if ($conn!=0) {
    echo "Connection ok!";
    $res=ovrimos_exec ($conn,"select table_id, table_name from sys.tables");
    if ($res != 0) {
        echo "Statement ok!";
        if (ovrimos_fetch_into ($res, $row)) {
            list ($table_id, $table_name) = $row;
            echo "table_id=".$table_id.", table_name=".$table_name."\n";
            if (ovrimos_fetch_into ($res, $row)) {
                list ($table_id, $table_name) = $row;
                echo "table_id=".$table_id.", table_name=".$table_name."\n";
            } else {
                echo "Next: error\n";
            }
        } else {
            echo "First: error\n";
        }
        ovrimos_free_result ($res);
    }
    ovrimos_close ($conn);
}
?>
```

This example will fetch a row.

ovrimos_fetch_row

(PHP 4 >= 4.0.3, PHP 5)

ovrimos_fetch_row -- Fetches a row from the result set

Description

int **ovrimos_fetch_row** (int result_id [, int how [, int row_number]])

ovrimos_fetch_row() is used to fetch a row from the result set.

ovrimos_fetch_row() fetches a row from the result set. Column values should be retrieved with other calls. Returns **TRUE** or **FALSE**.

Ejemplo 1. A fetch row example

```
<?php
$conn = ovrimos_connect ("remote.host", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res=ovrimos_exec ($conn, "select table_id, table_name from sys.tables");
    if ($res != 0) {
        echo "Statement ok!";
        if (ovrimos_fetch_row ($res, "First")) {
            $table_id = ovrimos_result ($res, 1);
            $table_name = ovrimos_result ($res, 2);
            echo "table_id=".$table_id.", table_name=".$table_name."\n";
            if (ovrimos_fetch_row ($res, "Next")) {
                $table_id = ovrimos_result ($res, "table_id");
                $table_name = ovrimos_result ($res, "table_name");
                echo "table_id=".$table_id.", table_name=".$table_name."\n";
            } else {
                echo "Next: error\n";
            }
        } else {
            echo "First: error\n";
        }
        ovrimos_free_result ($res);
    }
    ovrimos_close ($conn);
}
?>
```

This will fetch a row and print the result.

ovrimos_field_len

(PHP 4 >= 4.0.3, PHP 5)

ovrimos_field_len -- Returns the length of the output column

Description

int **ovrimos_field_len** (int result_id, int field_number)

ovrimos_field_len() is used to get the length of the output column.

ovrimos_field_len() returns the length of the output column at the (1-based) index specified.

ovrimos_field_name

(PHP 4 >= 4.0.3, PHP 5)

ovrimos_field_name -- Returns the output column name

Description

int **ovrimos_field_name** (int result_id, int field_number)

ovrimos_field_name() is used to get the output column name.

ovrimos_field_name() returns the output column name at the (1-based) index specified.

ovrimos_field_num

(PHP 4 >= 4.0.3, PHP 5)

ovrimos_field_num -- Returns the (1-based) index of the output column

Description

int **ovrimos_field_num** (int result_id, string field_name)

ovrimos_field_num() is used to get the (1-based) index of the output column.

ovrimos_field_num() returns the (1-based) index of the output column specified by name, or **`FALSE`**.

ovrimos_field_type

(PHP 4 >= 4.0.3, PHP 5)

ovrimos_field_type -- Returns the (numeric) type of the output column

Description

int **ovrimos_field_type** (int result_id, int field_number)

ovrimos_field_type() is used to get the (numeric) type of the output column.

ovrimos_field_type() returns the (numeric) type of the output column at the (1-based) index specified.

ovrimos_free_result

(PHP 4 >= 4.0.3, PHP 5)

ovrimos_free_result -- Frees the specified result_id

Description

int **ovrimos_free_result** (int result_id)

ovrimos_free_result() is used to free the result_id.

ovrimos_free_result() frees the specified result_id. Returns **TRUE**.

ovrimos_longreadlen

(PHP 4 >= 4.0.3, PHP 5)

ovrimos_longreadlen -- Specifies how many bytes are to be retrieved from long datatypes

Description

int **ovrimos_longreadlen** (int result_id, int length)

ovrimos_longreadlen() is used to specify how many bytes are to be retrieved from long datatypes.

ovrimos_longreadlen() specifies how many bytes are to be retrieved from long datatypes (long varchar and long varbinary). Default is zero. Regardless of its taking a result_id as an argument, it currently sets this parameter for all result sets. Returns **TRUE**.

ovrimos_num_fields

(PHP 4 >= 4.0.3, PHP 5)

ovrimos_num_fields -- Returns the number of columns

Description

int **ovrimos_num_fields** (int result_id)

ovrimos_num_fields() is used to get the number of columns.

ovrimos_num_fields() returns the number of columns in a result_id resulting from a query.

ovrimos_num_rows

(PHP 4 >= 4.0.3, PHP 5)

ovrimos_num_rows -- Returns the number of rows affected by update operations

Description

int **ovrimos_num_rows** (int result_id)

ovrimos_num_rows() is used to get the number of rows affected by update operations.

ovrimos_num_rows() returns the number of rows affected by update operations.

ovrimos_prepare

(PHP 4 >= 4.0.3, PHP 5)

ovrimos_prepare -- Prepares an SQL statement

Description

int **ovrimos_prepare** (int connection_id, string query)

ovrimos_prepare() is used to prepare an SQL statement.

ovrimos_prepare() prepares an SQL statement and returns a result_id (or **FALSE** on failure).

Ejemplo 1. Connect to Ovrimos SQL Server and prepare a statement

```
<?php
$conn=ovrimos_connect ("db_host", "8001", "admin", "password");
if ($conn!=0) {
    echo "Connection ok!";
    $res=ovrimos_prepare ($conn, "select table_id, table_name
                                  from sys.tables where table_id=1");
    if ($res != 0) {
        echo "Prepare ok!";
        if (ovrimos_execute ($res)) {
            echo "Execute ok!\n";
            ovrimos_result_all ($res);
        } else {
            echo "Execute not ok!";
        }
        ovrimos_free_result ($res);
    } else {
        echo "Prepare not ok!\n";
    }
    ovrimos_close ($conn);
}
?>
```

This will connect to Ovrimos SQL Server, prepare a statement and the execute it.

ovrimos_result_all

(PHP 4 >= 4.0.3, PHP 5)

ovrimos_result_all -- Prints the whole result set as an HTML table

Description

int **ovrimos_result_all** (int result_id [, string format])

ovrimos_result_all() is used to print an HTML table containing the whole result set.

ovrimos_result_all() prints the whole result set as an HTML table. Returns **TRUE** or **FALSE**.

Ejemplo 1. Prepare a statement, execute, and view the result

```
<?php
$conn = ovrimos_connect ("db_host", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res = ovrimos_prepare ($conn, "select table_id, table_name
                                    from sys.tables where table_id = 7");
    if ($res != 0) {
        echo "Prepare ok!";
        if (ovrimos_execute ($res, array(3))) {
            echo "Execute ok!\n";
            ovrimos_result_all ($res);
        } else {
            echo "Execute not ok!";
        }
        ovrimos_free_result ($res);
    } else {
        echo "Prepare not ok!\n";
    }
    ovrimos_close ($conn);
}
?>
```

This will execute an SQL statement and print the result in an HTML table.

Ejemplo 2. Ovrimos_result_all with meta-information

```
<?php
$conn = ovrimos_connect ("db_host", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res = ovrimos_exec ($conn, "select table_id, table_name
                                 from sys.tables where table_id = 1")
    if ($res != 0) {
        echo "Statement ok! cursor=".ovrimos_cursor ($res)."\n";
        $colnb = ovrimos_num_fields ($res);
        echo "Output columns=".$colnb."\n";
        for ($i=1; $i<=$colnb; $i++) {
            $name = ovrimos_field_name ($res, $i);
            $type = ovrimos_field_type ($res, $i);
            $len = ovrimos_field_len ($res, $i);
            echo "Column ".$i." name=".$name." type=".$type." len=".$len."\n";
        }
        ovrimos_result_all ($res);
        ovrimos_free_result ($res);
    }
    ovrimos_close ($conn);
}
?>
```

Ejemplo 3. ovrimos_result_all example

```
<?php
$conn = ovrimos_connect ("db_host", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res = ovrimos_exec ($conn, "update test set i=5");
    if ($res != 0) {
        echo "Statement ok!";
        echo ovrimos_num_rows ($res)." rows affected\n";
        ovrimos_free_result ($res);
    }
    ovrimos_close ($conn);
}
?>
```

ovrimos_result

(PHP 4 >= 4.0.3, PHP 5)

ovrimos_result -- Retrieves the output column

Description

int **ovrimos_result** (int result_id, mixed field)

ovrimos_result() is used to retrieve the output column.

ovrimos_result() retrieves the output column specified by 'field', either as a string or as an 1-based index.

ovrimos_rollback

(PHP 4 >= 4.0.3, PHP 5)

ovrimos_rollback -- Rolls back the transaction

Description

int **ovrimos_rollback** (int connection_id)

ovrimos_rollback() is used to roll back the transaction.

ovrimos_rollback() rolls back the transaction.

XCIV. Parsekit Functions

Introducción

These functions allow runtime analysis of opcodes compiled from PHP scripts.

Instalación

Esta extension PECL no esta ligada a PHP.

Mas informacion sobre nuevos lanzamientos, descargas ficheros de fuentes, informacion sobre los responsables asi como un 'CHANGELOG', se puede encontrar aqui: http://pecl.php.net/package/parsekit.

Podeis descargar esta DLL de las extensiones PECL desde la pagina PHP Downloads o desde http://snaps.php.net/.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

PARSEKIT_QUIET (int)

Return full detail, but without unnecessary NULL extries.

PARSEKIT_SIMPLE (int)

Return shorthand opcode notation.

PARSEKIT_EXTENDED_VALUE (int)

Opnode Flag

PARSEKIT_RESULT_CONST (int)

Opnode Flag

PARSEKIT_RESULT_EA_TYPE (int)

Opnode Flag

PARSEKIT_RESULT_JMP_ADDR (int)

Opnode Flag

PARSEKIT_RESULT_OPARRAY (int)

Opnode Flag

PARSEKIT_RESULT_OPLINE (int)

Opnode Flag

PARSEKIT_RESULT_VAR (int)

Opnode Flag

PARSEKIT_USAGE_UNKNOWN (int)

Opnode Flag

PARSEKIT_ZEND_INTERNAL_CLASS (int)

Class Type

PARSEKIT_ZEND_USER_CLASS (int)

Class Type

PARSEKIT_ZEND_EVAL_CODE (int)

Function Type

PARSEKIT_ZEND_INTERNAL_FUNCTION (int)

Function Type

PARSEKIT_ZEND_OVERLOADED_FUNCTION (int)

Function Type

PARSEKIT_ZEND_OVERLOADED_FUNCTION_TEMPORARY (int) PHP >= 5.0.0

Function Type

PARSEKIT_ZEND_USER_FUNCTION (int)

Function Type

PARSEKIT_IS_CONST (int)

Node Type

PARSEKIT_IS_TMP_VAR (int)

Node Type

PARSEKIT_IS_UNUSED (int)

Node Type

PARSEKIT_IS_VAR (int)

Node Type

PARSEKIT_ZEND_ADD (int)

Opcode

PARSEKIT_ZEND_ADD_ARRAY_ELEMENT (int)

Opcode

PARSEKIT_ZEND_ADD_CHAR (int)

Opcode

PARSEKIT_ZEND_ADD_INTERFACE (int) PHP >= 5.0.0

Opcode

PARSEKIT_ZEND_ADD_STRING (int)

Opcode

PARSEKIT_ZEND_ADD_VAR (int)

Opcode

PARSEKIT_ZEND_ASSIGN (int)

Opcode

PARSEKIT_ZEND_ASSIGN_ADD (int)

Opcode

PARSEKIT_ZEND_ASSIGN_BW_AND (int)

Opcode

PARSEKIT_ZEND_ASSIGN_BW_OR (int)

Opcode

PARSEKIT_ZEND_ASSIGN_BW_XOR (int)

Opcode

PARSEKIT_ZEND_ASSIGN_CONCAT (int)

Opcode

PARSEKIT_ZEND_ASSIGN_DIM (int) PHP >= 5.0.0

Opcode

PARSEKIT_ZEND_ASSIGN_DIV (int)

Opcode

PARSEKIT_ZEND_ASSIGN_MOD (int)

Opcode

PARSEKIT_ZEND_ASSIGN_MUL (int)

Opcode

PARSEKIT_ZEND_ASSIGN_OBJ (int) PHP >= 5.0.0

Opcode

PARSEKIT_ZEND_ASSIGN_REF (int)

Opcode

PARSEKIT_ZEND_ASSIGN_SL (int)

Opcode

PARSEKIT_ZEND_ASSIGN_SR (int)

Opcode

PARSEKIT_ZEND_ASSIGN_SUB (int)

Opcode

PARSEKIT_ZEND_BEGIN_SILENCE (int)

Opcode

PARSEKIT_ZEND_BOOL (int)

Opcode

PARSEKIT_ZEND_BOOL_NOT (int)

Opcode

PARSEKIT_ZEND_BOOL_XOR (int)

Opcode

PARSEKIT_ZEND_BRK (int)

Opcode

PARSEKIT_ZEND_BW_AND (int)

Opcode

PARSEKIT_ZEND_BW_NOT (int)

Opcode

PARSEKIT_ZEND_BW_OR (int)

Opcode

PARSEKIT_ZEND_BW_XOR (int)

Opcode

PARSEKIT_ZEND_CASE (int)

Opcode

PARSEKIT_ZEND_CAST (int)

Opcode

PARSEKIT_ZEND_CATCH (int) PHP >= 5.0.0

Opcode

PARSEKIT_ZEND_CLONE (int) PHP >= 5.0.0

Opcode

PARSEKIT_ZEND_CONCAT (int)

Opcode

PARSEKIT_ZEND_CONT (int)

Opcode

PARSEKIT_ZEND_DECLARE_CLASS (int) PHP >= 5.0.0

Opcode

PARSEKIT_ZEND_DECLARE_FUNCTION (int) PHP >= 5.0.0

Opcode

PARSEKIT_ZEND_DECLARE_INHERITED_CLASS (int) PHP >= 5.0.0

Opcode

PARSEKIT_ZEND_DIV (int)

Opcode

PARSEKIT_ZEND_DO_FCALL (int)

Opcode

PARSEKIT_ZEND_DO_FCALL_BY_NAME (int)

Opcode

PARSEKIT_ZEND_ECHO (int)

Opcode

PARSEKIT_ZEND_END_SILENCE (int)

Opcode

PARSEKIT_ZEND_EXIT (int)

Opcode

PARSEKIT_ZEND_EXT_FCALL_BEGIN (int)

Opcode

PARSEKIT_ZEND_EXT_FCALL_END (int)

Opcode

PARSEKIT_ZEND_EXT_NOP (int)

Opcode

PARSEKIT_ZEND_EXT_STMT (int)

Opcode

PARSEKIT_ZEND_FETCH_CLASS (int) PHP >= 5.0.0

Opcode

PARSEKIT_ZEND_FETCH_CONSTANT (int)

Opcode

PARSEKIT_ZEND_FETCH_DIM_FUNC_ARG (int)

Opcode

PARSEKIT_ZEND_FETCH_DIM_IS (int)

Opcode

PARSEKIT_ZEND_FETCH_DIM_R (int)

Opcode

PARSEKIT_ZEND_FETCH_DIM_RW (int)

Opcode

PARSEKIT_ZEND_FETCH_DIM_TMP_VAR (int)

Opcode

PARSEKIT_ZEND_FETCH_DIM_UNSET (int)

Opcode

PARSEKIT_ZEND_FETCH_DIM_W (int)

Opcode

PARSEKIT_ZEND_FETCH_FUNC_ARG (int)

Opcode

PARSEKIT_ZEND_FETCH_IS (int)

Opcode

PARSEKIT_ZEND_FETCH_OBJ_FUNC_ARG (int)

Opcode

PARSEKIT_ZEND_FETCH_OBJ_IS (int)

Opcode

PARSEKIT_ZEND_FETCH_OBJ_R (int)

Opcode

PARSEKIT_ZEND_FETCH_OBJ_RW (int)

Opcode

PARSEKIT_ZEND_FETCH_OBJ_UNSET (int)

Opcode

PARSEKIT_ZEND_FETCH_OBJ_W (int)

Opcode

PARSEKIT_ZEND_FETCH_R (int)

Opcode

PARSEKIT_ZEND_FETCH_RW (int)

Opcode

PARSEKIT_ZEND_FETCH_UNSET (int)

Opcode

PARSEKIT_ZEND_FETCH_W (int)

Opcode

PARSEKIT_ZEND_FE_FETCH (int)

Opcode

PARSEKIT_ZEND_FE_RESET (int)

Opcode

PARSEKIT_ZEND_FREE (int)

Opcode

PARSEKIT_ZEND_HANDLE_EXCEPTION (int) PHP >= 5.0.0

Opcode

PARSEKIT_ZEND_IMPORT_CLASS (int) PHP >= 5.0.0

Opcode

PARSEKIT_ZEND_IMPORT_CONST (int) PHP >= 5.0.0

Opcode

PARSEKIT_ZEND_IMPORT_FUNCTION (int) PHP >= 5.0.0

Opcode

PARSEKIT_ZEND_INCLUDE_OR_EVAL (int)

Opcode

PARSEKIT_ZEND_INIT_ARRAY (int)

Opcode

PARSEKIT_ZEND_INIT_CTOR_CALL (int) PHP >= 5.0.0

Opcode

PARSEKIT_ZEND_INIT_FCALL_BY_NAME (int)

Opcode

PARSEKIT_ZEND_INIT_METHOD_CALL (int) PHP >= 5.0.0

Opcode

PARSEKIT_ZEND_INIT_STATIC_METHOD_CALL (int) PHP >= 5.0.0

Opcode

PARSEKIT_ZEND_INIT_STRING (int)

Opcode

PARSEKIT_ZEND_INSTANCEOF (int) PHP >= 5.0.0

Opcode

PARSEKIT_ZEND_ISSET_ISEMPTY (int) PHP < 5.0.0

Opcode

PARSEKIT_ZEND_ISSET_ISEMPTY_DIM_OBJ (int) PHP >= 5.0.0

Opcode

PARSEKIT_ZEND_ISSET_ISEMPTY_PROP_OBJ (int) PHP >= 5.0.0

Opcode

PARSEKIT_ZEND_ISSET_ISEMPTY_VAR (int) PHP >= 5.0.0

Opcode

PARSEKIT_ZEND_IS_EQUAL (int)

Opcode

PARSEKIT_ZEND_IS_IDENTICAL (int)

Opcode

PARSEKIT_ZEND_IS_NOT_EQUAL (int)

Opcode

PARSEKIT_ZEND_IS_NOT_IDENTICAL (int)

Opcode

PARSEKIT_ZEND_IS_SMALLER (int)

Opcode

PARSEKIT_ZEND_IS_SMALLER_OR_EQUAL (int)

Opcode

PARSEKIT_ZEND_JMP (int)

Opcode

PARSEKIT_ZEND_JMPNZ (int)

Opcode

PARSEKIT_ZEND_JMPNZ_EX (int)

Opcode

PARSEKIT_ZEND_JMPZ (int)

Opcode

PARSEKIT_ZEND_JMPZNZ (int)

Opcode

PARSEKIT_ZEND_JMPZ_EX (int)

Opcode

PARSEKIT_ZEND_JMP_NO_CTOR (int)

Opcode

PARSEKIT_ZEND_MOD (int)

Opcode

PARSEKIT_ZEND_MUL (int)

Opcode

PARSEKIT_ZEND_NEW (int)

Opcode

PARSEKIT_ZEND_NOP (int)

Opcode

PARSEKIT_ZEND_OP_DATA (int) PHP >= 5.0.0

Opcode

PARSEKIT_ZEND_POST_DEC (int)

Opcode

PARSEKIT_ZEND_POST_DEC_OBJ (int) PHP >= 5.0.0

Opcode

PARSEKIT_ZEND_POST_INC (int)

Opcode

PARSEKIT_ZEND_POST_INC_OBJ (int) PHP >= 5.0.0

Opcode

PARSEKIT_ZEND_PRE_DEC (int)

Opcode

PARSEKIT_ZEND_PRE_DEC_OBJ (int) PHP >= 5.0.0

Opcode

PARSEKIT_ZEND_PRE_INC (int)

Opcode

PARSEKIT_ZEND_PRE_INC_OBJ (int) PHP >= 5.0.0

Opcode

PARSEKIT_ZEND_PRINT (int)

Opcode

PARSEKIT_ZEND_QM_ASSIGN (int)

Opcode

PARSEKIT_ZEND_RAISE_ABSTRACT_ERROR (int) PHP >= 5.0.0

Opcode

PARSEKIT_ZEND_RECV (int)

Opcode

PARSEKIT_ZEND_RECV_INIT (int)

Opcode

PARSEKIT_ZEND_RETURN (int)

Opcode

PARSEKIT_ZEND_SEND_REF (int)

Opcode

PARSEKIT_ZEND_SEND_VAL (int)

Opcode

PARSEKIT_ZEND_SEND_VAR (int)

Opcode

PARSEKIT_ZEND_SEND_VAR_NO_REF (int)

Opcode

PARSEKIT_ZEND_SL (int)

Opcode

PARSEKIT_ZEND_SR (int)

Opcode

PARSEKIT_ZEND_SUB (int)

Opcode

PARSEKIT_ZEND_SWITCH_FREE (int)

Opcode

PARSEKIT_ZEND_THROW (int) PHP >= 5.0.0

Opcode

PARSEKIT_ZEND_TICKS (int)

Opcode

PARSEKIT_ZEND_UNSET_DIM_OBJ (int)

Opcode

PARSEKIT_ZEND_UNSET_VAR (int)

Opcode

PARSEKIT_ZEND_VERIFY_ABSTRACT_CLASS (int) PHP >= 5.0.0

Opcode

Tabla de contenidos

parsekit_compile_file

(no version information, might be only in CVS)

parsekit_compile_file -- Compile a string of PHP code and return the resulting op array

Descripción

array **parsekit_compile_file** (string filename [, array &errors [, int options]])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Lista de parámetros

filename

A string containing the name of the file to compile. Similar to the argument to **include()**.

errors

A 2D hash of errors (including fatal errors) encountered during compilation. Returned by reference.

options

One of either **PARSEKIT_QUIET** or **PARSEKIT_SIMPLE**. To produce varying degrees of verbosity in the returned output.

Valores retornados

Returns a complex multi-layer array structure as detailed below.

Ejemplos

Ejemplo 1. parsekit_compile_file() example

```
<?php
var_dump(parsekit_compile_file('hello_world.php', $errors, PARSEKIT_SIMPLE));
?>
```

El resultado del ejemplo seria:

```
array(5) {
  [0]=>
  string(37) "ZEND_ECHO UNUSED 'Hello World' UNUSED"
  [1]=>
  string(30) "ZEND_RETURN UNUSED NULL UNUSED"
  [2]=>
  string(42) "ZEND_HANDLE_EXCEPTION UNUSED UNUSED UNUSED"
  ["function_table"]=>
  NULL
  ["class_table"]=>
  NULL
}
```

Ver también

parsekit_compile_string()

parsekit_compile_string

(no version information, might be only in CVS)

parsekit_compile_string -- Compile a string of PHP code and return the resulting op array

Descripción

array **parsekit_compile_string** (string phpcode [, array &errors [, int options]])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Lista de parámetros

phpcode

A string containing phpcode. Similar to the argument to **eval()**.

errors

A 2D hash of errors (including fatal errors) encountered during compilation. Returned by reference.

options

One of either **PARSEKIT_QUIET** or **PARSEKIT_SIMPLE**. To produce varying degrees of verbosity in the returned output.

Valores retornados

Returns a complex multi-layer array structure as detailed below.

Ejemplos

Ejemplo 1. parsekit_compile_string() example

```
<?php
  $ops = parsekit_compile_string('
echo "Foo\n";
', $errors, PARSEKIT_QUIET);

  var_dump($ops);
?>
```

El resultado del ejemplo seria:

```
array(20) {
  ["type"]=>
  int(4)
  ["type_name"]=>
  string(14) "ZEND_EVAL_CODE"
  ["fn_flags"]=>
  int(0)
  ["num_args"]=>
  int(0)
  ["required_num_args"]=>
  int(0)
  ["pass_rest_by_reference"]=>
  bool(false)
  ["uses_this"]=>
  bool(false)
  ["line_start"]=>
  int(0)
  ["line_end"]=>
  int(0)
  ["return_reference"]=>
  bool(false)
  ["refcount"]=>
  int(1)
  ["last"]=>
  int(3)
  ["size"]=>
  int(3)
  ["T"]=>
  int(0)
  ["last_brk_cont"]=>
  int(0)
  ["current_brk_cont"]=>
  int(-1)
  ["backpatch_count"]=>
  int(0)
  ["done_pass_two"]=>
  bool(true)
  ["filename"]=>
  string(17) "Parsekit Compiler"
  ["opcodes"]=>
  array(3) {
    [8594800]=>
    array(5) {
      ["opcode"]=>
      int(40)
      ["opcode_name"]=>
      string(9) "ZEND_ECHO"
      ["flags"]=>
      int(768)
      ["op1"]=>
      array(3) {
        ["type"]=>
        int(1)
        ["type_name"]=>
        string(8) "IS_CONST"
        ["constant"]=>
        &string(4) "Foo
"
      }
      ["lineno"]=>
      int(2)
    }
    ["859484C"]=>
    array(6) {
      ["opcode"]=>
      int(62)
      ["opcode_name"]=>
      string(11) "ZEND_RETURN"
      ["flags"]=>
      int(16777984)
      ["op1"]=>
      array(3) {
```

Ver también

parsekit_compile_file()

parsekit_func_arginfo

(no version information, might be only in CVS)

parsekit_func_arginfo -- Return information regarding function argument(s)

Descripción

array **parsekit_func_arginfo** (mixed function)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Lista de parámetros

function

A string describing a function, or an array describing a class/method.

Valores retornados

Returns an array containing argument information.

Ejemplos

Ejemplo 1. parsekit_func_arginfo() example

```
<?php
function foo($bar, stdClass $baz, &$bomb, $bling = false) {
}

var_dump(parsekit_func_arginfo('foo'));
?>
```

El resultado del ejemplo seria:

```
array(4) {
  [0]=>
  array(3) {
    ["name"]=>
    string(3) "bar"
    ["allow_null"]=>
    bool(true)
    ["pass_by_reference"]=>
    bool(false)
  }
  [1]=>
  array(4) {
    ["name"]=>
    string(3) "baz"
    ["class_name"]=>
    string(8) "stdClass"
    ["allow_null"]=>
    bool(false)
    ["pass_by_reference"]=>
    bool(false)
  }
  [2]=>
  array(3) {
    ["name"]=>
    string(4) "bomb"
    ["allow_null"]=>
    bool(true)
    ["pass_by_reference"]=>
    bool(true)
  }
  [3]=>
  array(3) {
    ["name"]=>
    string(5) "bling"
    ["allow_null"]=>
    bool(true)
    ["pass_by_reference"]=>
    bool(false)
  }
}
```

XCV. Funciones de Control de Procesos

Introducción

El soporte de Control de Procesos en PHP implementa el estilo Unix de creaciÃ, n de procesos, ejecuciÃ, n de programa, administraciÃ, n de señales y finalizaciÃ, n de procesos. El Control de Procesos no debería estar activado para un servidor de entorno web, ya que podrían ocurrir resultados inesperados utilizando las funciones de Control de Procesos.

Esta documentaciÃ, n explica en modo general como se utilizan las funciones de Control de Procesos. Para una informaciÃ, n más detallada sobre controles de procesos en Unix deberías acudir a la documentaciÃ, n de tu sistema sobre fork(2), waitpid(2), signal(2) o al manual de referecia de programaciÃ, n avanzada bajo entornos Unix (Advanced Programming in the UNIX Environment) de Richard Stevens, Addison-Wesley.

PCNTL ahora utiliza señales para manejar el mecanismo de llamadas de retorno, que es mucho más rápido que el anterior mecanismo. Este cambio sigue la misma semántica que utilizar "señales del usuario". Debes utilizar **declare()** para definir que lugares en tus programas se permiten las llamadas de retorno. De esta manera se minimiza la carga de eventos asíncronos. Anteriormente, al compilar PHP con la extensiÃ, n pcntl siempre estaban activadas las llamadas de retorno, produciendo así una carga innecesaria para las aplicaciones que no utilizaban pcntl.

Para scripts pcntl anteriores al PHP 4.3.0 hay que realizar los ajustes para definir con el **declare()** en que secciones de la aplicaciÃ, n se permiten las llamadas de retorno o simplemente permitirlo en toda la aplicaciÃ, n mediante la sintaxis global de **declare()**.

> **Nota:** Esta extensión no está disponible en plataformas Windows

---

Requirimientos

No se necesitan bibliotecas externas para construir esta extensión

---

Instalación

El soporte de Control de Procesos en PHP no está activado por defecto. Para activarlo, tienes que compilar la versión CGI o CLI de PHP con la opción *--enable-pcntl* en la configuración.

> **Nota:** Actualmente este modulo no funciona en plataformas diferentes a Unix (Windows).

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Las siguiente lista de señales están soportadas por las funciones de Control de Procesos. Comprueba el manual de señales de tu sistema para más detalles sobre el comportamiento por defecto de las mismas.

`WNOHANG` (**integer**)

`WUNTRACED` (**integer**)

`SIG_IGN` (**integer**)

`SIG_DFL` (**integer**)

`SIG_ERR` (**integer**)

`SIGHUP` (**integer**)

SIGINT (integer)

SIGQUIT (integer)

SIGILL (integer)

SIGTRAP (integer)

SIGABRT (integer)

SIGIOT (integer)

SIGBUS (integer)

SIGFPE (integer)

SIGKILL (integer)

SIGUSR1 (integer)

SIGSEGV (integer)

SIGUSR2 (integer)

SIGPIPE (integer)

SIGALRM (integer)

SIGTERM (integer)

SIGSTKFLT (integer)

SIGCLD (integer)

SIGCHLD (integer)

SIGCONT (integer)

SIGSTOP (integer)

SIGTSTP (integer)

SIGTTIN (integer)

SIGTTOU (integer)

SIGURG (integer)

SIGXCPU (integer)

SIGXFSZ (integer)

SIGVTALRM (integer)

SIGPROF (integer)

SIGWINCH (integer)

SIGPOLL (integer)

SIGIO (integer)

SIGPWR (integer)

SIGSYS (integer)

SIGBABY (integer)

Ejemplos

Este ejemplo realiza un fork (bifurcaciÃ, n) de un proceso daemon (demonio) con administraciÃ, n de señales.

Ejemplo 1. Ejemplo de Control de Procesos

```php
<?php
declare(ticks=1);

$pid = pcntl_fork();
if ($pid == -1) {
     die("no se puede hacer fork");
} else if ($pid) {
     exit(); // somos el proceso padre
} else {
     // somos el proceso hijo
}

// detatch desde la terminal
if (!posix_setsid()) {
    die("no se puede hacer un detach desde la terminal");
}

// bucle infinito realizando tareas
while (1) {

    // hacer algo interesante aquí ..

}

function sig_handler($signo)
{

     switch ($signo) {
         case SIGTERM:
             // tareas de finalización
             exit;
             break;
         case SIGHUP:
             // tareas de reinicio
             break;
         default:
             // tareas para las demás señales
     }

}

// congiguración de las señales
pcntl_signal(SIGTERM, "sig_handler");
pcntl_signal(SIGHUP, "sig_handler");

?>
```

Ver también

Te puede ser útil echar un vistazo a POSIX functions

Tabla de contenidos

pcntl_wifstopped -- Returns **TRUE** if child process is currently stopped
pcntl_wstopsig -- Returns the signal which caused the child to stop
pcntl_wtermsig -- Returns the signal which caused the child to terminate

pcntl_alarm

(PHP 4 >= 4.3.0, PHP 5)

pcntl_alarm -- Programa una alarma para hacer una llamada de una señal

Descripción

int **pcntl_alarm** (int segundos)

La función **pcntl_alarm()** crea un contador para enviar una señal *SIGALRM*. Si *segundos* es cero, no se crea ninguna alarma. Cualquier llamada a **pcntl_alarm()** cancelará las alarmas anteriormente programadas.

pcntl_alarm() devolverá el número de segundos restantes para la ejecución de la alarma programada, o *0* si no existe ninguna alarma programada.

pcntl_exec

(PHP 4 >= 4.2.0, PHP 5)

pcntl_exec -- Ejecuta un programa específico en el espacio de proceso actual

Descripción

bool **pcntl_exec** (string ruta [, array args [, array envs]])

pcntl_exec() ejecuta el programa *path* con los argumentos *args*. *path* debe ser un binario ejecutable o un script apuntando a un ejecutable con la ruta correcta en la primera línea (por ejemplo #!/usr/local/bin/perl). Para más información mira la página del man execve(2) en tu sistema.

args debe ser una matriz con los argumentos que se le pasan al programa.

envs debe ser una matriz de cadenas de texto en la cual se pasan variables de entorno al programa. La matriz debe tener el formato indice => contenido, el indice seráa el nombre de la variable de entorno y el contenido será el valor de la variable.

pcntl_exec() devuelve **FALSE** en errores.

pcntl_fork

(PHP 4 >= 4.1.0, PHP 5)

pcntl_fork -- Forks the currently running process

Description

int **pcntl_fork** (void)

The **pcntl_fork()** function creates a child process that differs from the parent process only in its PID and PPID. Please see your system's fork(2) man page for specific details as to how fork works on your system.

On success, the PID of the child process is returned in the parent's thread of execution, and a 0 is returned in the child's thread of execution. On failure, a -1 will be returned in the parent's context, no child process will be created, and a PHP error is raised.

Ejemplo 1. pcntl_fork() example

```
<?php

$pid = pcntl_fork();
if ($pid == -1) {
     die('could not fork');
} else if ($pid) {
     // we are the parent
     pcntl_wait($status); //Protect against Zombie children
} else {
     // we are the child
}

?>
```

See also **pcntl_waitpid()** and **pcntl_signal()**.

pcntl_getpriority

(PHP 5)

pcntl_getpriority -- Get the priority of any process

Description

int **pcntl_getpriority** ([int pid [, int process_identifier]])

pcntl_getpriority() gets the priority of *pid*. If *pid* is not specified, the pid of the current process is used. Because priority levels can differ between system types and kernel versions, please see your system's getpriority(2) man page for specific details.

pcntl_getpriority() returns the priority of the process or **FALSE** on error. A lower numerical value causes more favorable scheduling.

process_identifier is one of **PRIO_PGRP**, **PRIO_USER** or **PRIO_PROCESS**.

Aviso
Esta función puede devolver **FALSE**, pero también puede devolver un valor no-booleano que será evaluado **FALSE**, como por ejemplo *0* o "". Por favor, lea la sección Booleans para más información. Utilice el operador === para comprobar el valor devuelto por esta función.

pcntl_setpriority

(PHP 5)

pcntl_setpriority -- Change the priority of any process

Description

bool **pcntl_setpriority** (int priority [, int pid [, int process_identifier]])

pcntl_setpriority() sets the priority of *pid* to *priority*. If *pid* is not specified, the pid of the current process is used.

priority is generally a value in the range *-20* to *20*. The default priority is *0* while a lower numerical value causes more favorable scheduling. Because priority levels can differ between system types and kernel versions, please see your system's setpriority(2) man page for specific details.

process_identifier is one of **PRIO_PGRP**, **PRIO_USER** or **PRIO_PROCESS**.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

pcntl_signal

(PHP 4 >= 4.1.0, PHP 5)

pcntl_signal -- Installs a signal handler

Description

bool **pcntl_signal** (int signo, callback handle [, bool restart_syscalls])

The **pcntl_signal()** function installs a new signal handler for the signal indicated by *signo*. The signal handler is set to *handler* which may be the name of a user created function, or either of the two global constants SIG_IGN or SIG_DFL. The optional *restart_syscalls* specifies whether system call restarting should be used when this signal arrives and defaults to **TRUE**.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

> **Nota:** The optional *restart_syscalls* parameter became available in PHP 4.3.0.
>
> **Nota:** The ability to use an object method as a callback became available in PHP 4.3.0. Note that when you set a handler to an object method, that object's reference count is increased which makes it persist until you either change the handler to something else, or your script ends.

Ejemplo 1. pcntl_signal() example

```
<?php
// tick use required as of PHP 4.3.0
declare(ticks = 1);

// signal handler function
function sig_handler($signo)
{

     switch ($signo) {
         case SIGTERM:
             // handle shutdown tasks
             exit;
             break;
         case SIGHUP:
             // handle restart tasks
             break;
         case SIGUSR1:
             echo "Caught SIGUSR1...\n";
             break;
         default:
             // handle all other signals
     }

}

echo "Installing signal handler...\n";

// setup signal handlers
pcntl_signal(SIGTERM, "sig_handler");
pcntl_signal(SIGHUP,  "sig_handler");
pcntl_signal(SIGUSR1, "sig_handler");

// or use an object, available as of PHP 4.3.0
// pcntl_signal(SIGUSR1, array($obj, "do_something");

echo"Generating signal SIGTERM to self...\n";

// send SIGUSR1 to current process id
posix_kill(posix_getpid(), SIGUSR1);

echo "Done\n"

?>
```

> **Nota:** As of PHP 4.3.0 PCNTL uses ticks as the signal handle callback mechanism, which is much faster than the previous mechanism. This change follows the same semantics as using "user ticks". You must use the declare() statement to specify the locations in your program where callbacks are allowed to occur for the signal handler to function properly (as used in the above example).

See also **pcntl_fork()** and **pcntl_waitpid()**.

pcntl_wait

(PHP 5)

pcntl_wait -- Waits on or returns the status of a forked child

Description

int **pcntl_wait** (int &status [, int options])

The wait function suspends execution of the current process until a child has exited, or until a signal

is delivered whose action is to terminate the current process or to call a signal handling function. If a child has already exited by the time of the call (a so-called "zombie" process), the function returns immediately. Any system resources used by the child are freed. Please see your system's wait(2) man page for specific details as to how wait works on your system.

pcntl_wait() returns the process ID of the child which exited, -1 on error or zero if WNOHANG was provided as an option (on wait3-available systems) and no child was available.

If wait3 is available on your system (mostly BSD-style systems), you can provide the optional *options* parameter. If this parameter is not provided, wait will be used for the system call. If wait3 is not available, providing a value for *options* will have no effect. The value of *options* is the value of zero or more of the following two constants *OR*'ed together:

Tabla 1. Possible values for *options* if wait3 is available

WNOHANG	Return immediately if no child has exited.
WUNTRACED	Return for children which are stopped, and whose status has not been reported.

pcntl_wait() will store status information in the *status* parameter which can be evaluated using the following functions: **pcntl_wifexited()**, **pcntl_wifstopped()**, **pcntl_wifsignaled()**, **pcntl_wexitstatus()**, **pcntl_wtermsig()** and **pcntl_wstopsig()**.

> **Nota:** This function is ñalent to calling **pcntl_waitpid()** with a *-1 pid* and no *options*.

See also **pcntl_fork()**, **pcntl_signal()**, **pcntl_wifexited()**, **pcntl_wifstopped()**, **pcntl_wifsignaled()**, **pcntl_wexitstatus()**, **pcntl_wtermsig()**, **pcntl_wstopsig()** and **pcntl_waitpid()**.

pcntl_waitpid

(PHP 4 >= 4.1.0, PHP 5)

pcntl_waitpid -- Waits on or returns the status of a forked child

Description

int **pcntl_waitpid** (int pid, int &status [, int options])

The **pcntl_waitpid()** function suspends execution of the current process until a child as specified by the *pid* argument has exited, or until a signal is delivered whose action is to terminate the current process or to call a signal handling function. If a child as requested by *pid* has already exited by the time of the call (a so-called "zombie" process), the function returns immediately. Any system resources used by the child are freed. Please see your system's waitpid(2) man page for specific details as to how waitpid works on your system.

pcntl_waitpid() returns the process ID of the child which exited, -1 on error or zero if WNOHANG was used and no child was available

The value of *pid* can be one of the following:

Tabla 1. possible values for *pid*

< *-1*	wait for any child process whose process group ID is equal to the absolute value of *pid*.
-1	wait for any child process; this is the same behaviour that the wait function exhibits.
0	wait for any child process whose process group ID is equal to that of the calling process.
> 0	wait for the child whose process ID is equal to the value of *pid*.

Nota: Specifying *-1* as the *pid* is ñalent to the functionality **pcntl_wait()** provides (minus *options*).

pcntl_waitpid() will store status information in the *status* parameter which can be evaluated using the following functions: **pcntl_wifexited()**, **pcntl_wifstopped()**, **pcntl_wifsignaled()**, **pcntl_wexitstatus()**, **pcntl_wtermsig()** and **pcntl_wstopsig()**.

The value of *options* is the value of zero or more of the following two global constants *OR*'ed together:

Tabla 2. possible values for *options*

WNOHANG	return immediately if no child has exited.
WUNTRACED	return for children which are stopped, and whose status has not been reported.

See also **pcntl_fork()**, **pcntl_signal()**, **pcntl_wifexited()**, **pcntl_wifstopped()**, **pcntl_wifsignaled()**, **pcntl_wexitstatus()**, **pcntl_wtermsig()** and **pcntl_wstopsig()**.

pcntl_wexitstatus

(PHP 4 >= 4.1.0, PHP 5)

pcntl_wexitstatus -- Returns the return code of a terminated child

Description

int **pcntl_wexitstatus** (int status)

Returns the return code of a terminated child. This function is only useful if **pcntl_wifexited()** returned **`TRUE`**.

El parametro *status* es un parametro de estado enviado a una llamada con exito de **pcntl_waitpid()**.

See also **pcntl_waitpid()** and **pcntl_wifexited()**.

pcntl_wifexited

(PHP 4 >= 4.1.0, PHP 5)

pcntl_wifexited -- Returns **`TRUE`** if status code represents a successful exit

Description

int **pcntl_wifexited** (int status)

Returns **TRUE** if the child status code represents a successful exit.

El parametro *status* es un parametro de estado enviado a una llamada con exito de **pcntl_waitpid()**.

See also **pcntl_waitpid()** and **pcntl_wexitstatus()**.

pcntl_wifsignaled

(PHP 4 >= 4.1.0, PHP 5)

pcntl_wifsignaled -- Returns **TRUE** if status code represents a termination due to a signal

Description

int **pcntl_wifsignaled** (int status)

Returns **TRUE** if the child process exited because of a signal which was not caught.

El parametro *status* es un parametro de estado enviado a una llamada con exito de **pcntl_waitpid()**.

See also **pcntl_waitpid()** and **pcntl_signal()**.

pcntl_wifstopped

(PHP 4 >= 4.1.0, PHP 5)

pcntl_wifstopped -- Returns **TRUE** if child process is currently stopped

Description

int **pcntl_wifstopped** (int status)

Returns **TRUE** if the child process which caused the return is currently stopped; this is only possible if the call to **pcntl_waitpid()** was done using the option *WUNTRACED*.

El parametro *status* es un parametro de estado enviado a una llamada con exito de **pcntl_waitpid()**.

See also **pcntl_waitpid()**.

pcntl_wstopsig

(PHP 4 >= 4.1.0, PHP 5)

pcntl_wstopsig -- Returns the signal which caused the child to stop

Description

int **pcntl_wstopsig** (int status)

Returns the number of the signal which caused the child to stop. This function is only useful if **pcntl_wifstopped()** returned **TRUE**.

El parametro *status* es un parametro de estado enviado a una llamada con exito de **pcntl_waitpid()**.

See also **pcntl_waitpid()** and **pcntl_wifstopped()**.

pcntl_wtermsig

(PHP 4 >= 4.1.0, PHP 5)

pcntl_wtermsig -- Returns the signal which caused the child to terminate

Description

int **pcntl_wtermsig** (int status)

Returns the number of the signal that caused the child process to terminate. This function is only useful if **pcntl_wifsignaled()** returned **TRUE**.

El parametro *status* es un parametro de estado enviado a una llamada con exito de **pcntl_waitpid()**.

See also **pcntl_waitpid()**, **pcntl_signal()** and **pcntl_wifsignaled()**.

XCVI. Funciones de Expresiones Regulares (Compatibles con Perl)

Introducción

La sintaxis para los patrones usados en éstas funciones se asemeja considerablemente con la sintaxis de Perl. La expresión debe estar rodeada por delimitadores, una barra acostada (/), por ejemplo. Cualquier caracter puede ser usado como delimitador siempre y cuando no sea alfanumérico ni la barra invertida (\). Si el caracter delimitador tiene que ser usado en la expresión misma, necesita ser escapado por la barra invertida. A partir de PHP 4.0.4, puede usar también los delimitadores de coincidencia tipo Perl (), {}, [], y <>. Vea Sintaxis de los Patrones para una explicación detallada.

El delimitador de cierre puede estar seguido de varios modificadores que afectan las coincidencias. Vea Modificadores de Patrón.

PHP soporta también expresiones regulares usando una sintaxis POSIX-extendida, por medio de las funciones regex POSIX-extendidas.

Aviso
Es importante que conozca sobre las limitaciones de PCRE. Lea http://www.pcre.org/pcre.txt para más información.

Requirimientos

El soporte de expresiones regulares es provisto por el paquete de biblioteca PCRE, el cual es software de código abierto, escrito por Philip Hazel, y con copyright de la Universidad de Cambridge, Inglaterra. Se encuentra disponible en ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/.

Instalación

A partir de PHP 4.2.0, estas funciones están habilitadas por defecto. Puede deshabilitar las funciones pcre con *--without-pcre-regex*. Use *--with-pcre-regex=DIR* para especificar la ubicación de los archivos de inclusión y bibliotecas de PCRE, si no desea usar la biblioteca incluida. En versiones más antiguas, usted tendrá que configurar y compilar PHP con *--with-pcre-regex[=DIR]* para poder hacer uso de estas funciones.

La versión para Windows de *PHP* tiene soporte nativo para esta extensión. No se necesita cargar ninguna extensión adicional para usar estas funciones.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

Tabla 1. constantes PREG

constante	descripción
PREG_PATTERN_ORDER	Ordena los resultados de modo que $coincidencias[0] sea una matriz de coincidencias del patrón completo, $coincidencias[1] sea una matriz de cadenas que coincidieron con el primer subpatrón entre paréntesis, y así sucesivamente. Esta bandera sólo es usada con **preg_match_all()**.

constante	descripción
PREG_SET_ORDER	Ordena los resultados de modo que $coincidencias[0] resulte ser una matriz del primer conjunto de coincidencias, $matches[1] sea una matriz del segundo conjunto de coincidencias, y así sucesivamente. Esta bandera únicamente es usada con **preg_match_all()**.
PREG_OFFSET_CAPTURE	Consulte la descripción de **PREG_SPLIT_OFFSET_CAPTURE**. Esta bandera está disponible desde PHP 4.3.0.
PREG_SPLIT_NO_EMPTY	Esta bandera le dice a **preg_split()** que devuelva únicamente resultados que no sean vacíos.
PREG_SPLIT_DELIM_CAPTURE	Esta bandera le indica a **preg_split()** que capture las expresiones entre paréntesis dentro del patrón de delimitación también. Esta bandera está disponible desde PHP 4.0.5.
PREG_SPLIT_OFFSET_CAPTURE	Si esta bandera está activa, la posición de desplazamiento correspondiente a cada coincidencia será devuelta también. Note que esto modifica el valor devuelto a una matriz en la que cada elemento es también una matriz que consiste de la cadena coincidente en el subíndice 0 y su posición de desplazamiento al interior de la cadena de asunto en el subíndice 1. Esta bandera está disponible a partir de PHP 4.3.0 y sólo es usada por **preg_split()**.

Ejemplos

Ejemplo 1. Ejemplos de patrones válidos

- */<\/\w+>/*
- *|(\d{3})-\d+|Sm*
- */^(?i)php[34]/*
- *{^\s+(\s+)?$}*

Ejemplo 2. Ejemplos de patrones inválidos

- */href='(.*)'* - carece de delimitador de cierre
- */\w+\s*\w+/J* - modificador 'J' desconocido
- *1-\d3-\d3-\d4|* - carece del delimitador de apertura

Tabla de contenidos

Modificadores de Patrón

Modificadores de Patrón -- Describe los posibles modificadores en patrones de expresiones regulares

Descripción

Los modificadores PCRE disponibles en la actualidad son listados a continuación. Los nombres entre paréntesis se refieren a nombres internos de PCRE para dichos modificadores.

i (PCRE_CASELESS)

Si éste modificador es definido, las letras en el patrón coincidirán tanto con letras mayúsculas como minúsculas.

m (PCRE_MULTILINE)

Por defecto, PCRE trata la cadena de asunto como si consistiera de una "única" línea de caracteres (aun si en realidad contiene varias). El meta-caracter de "inicio de línea" (^) coincide sólo al principio de la cadena, mientras que el meta-caracter de "fin de línea" ($) coincide sólo el final de la cadena, o antes un caracter de nueva línea final (a menos que el modificador *D* sea definido). Esto es igual que en Perl.

Cuando este modificador es definido, los constructores de "inicio de línea" y "fin de línea" coinciden inmediatamente después o inmediatamente antes de cualquier caracter de nueva línea, respectivamente, al igual que al comienzo o final absoluto de la cadena. Este comportamiento es ñalente al modificador /m del Perl. Si no hay caracteres "\n" en la cadena de asunto, o no hay ocurrencias de ^ o $ en un patrón, este modificador no tiene efecto alguno.

s (PCRE_DOTALL)

Si se define éste modificador, un meta-caracter de punto en el patrón coincidirá con todos los caracteres, incluyendo el de nueva línea. Sin él, los saltos de línea son excluídos. Este modificador es ñalente a /s en Perl. Una clase negativa como [^a] siempre coincide con un caracter de nueva línea, independientemente del uso de este modificador.

x (PCRE_EXTENDED)

Si éste modificador es definido, los caracteres de datos que representan espacios en blanco en el patrón son completamente ignorados, excepto cuando son escapados o cuando se encuentran al interior de una clase caracter, y los caracteres entre un # sin escapar fuera de una clase de caracter y el siguiente caracter de nueva línea, inclusive, son ignorados también. Esto es ñalente al modificador /x de Perl y hace posible incluir comentarios al interior de patrones complicados. Note, sin embargo, que esto es sólo aplicable a caracteres de datos. Los caracteres de espacio en blanco nunca pueden aparecer en secuencias de caracteres especiales en un patrón, por ejemplo al interior de la secuencia (?(, la cual inicia un sub-patrón condicional.

e

Si éste modificador es usado, **preg_replace()** realiza las sustituciones normales de referencias hacia atrás en la cadena de reemplazo, evalúa ésta como código PHP y usa el resultado para reemplazar la cadena de búsqueda. Las comillas sencillas y dobles son escapadas con barras invertidas en las referencias hacia atrás sustituidas.

Sólo **preg_replace()** usa éste modificador; es ignorado por otras funciones de PCRE.

Nota: Este modificador no se encontraba disponible en PHP 3.

A (PCRE_ANCHORED)

Si éste modificador es definido, el patrón es obligado a ser "anclado", es decir, es limitado para que coincida sólo al inicio de la cadena que está siendo analizada (la "cadena de asunto"). Este efecto puede alcanzarse también mediante las construcciones apropiadas en el patrón mismo, la cual es la única manera de hacerlo en Perl.

D (PCRE_DOLLAR_ENDONLY)

Si éste modificador es definido, un meta-caracter de signo dólar en el patrón coincide únicamente al final de la cadena de asunto. Sin éste modificador, un dólar coincide también inmediatamente antes del caracter final si éste es un salto de línea (pero no antes de cualquier otra nueva línea). Este modificador es ignorado si *m* es definido. No hay ñalente en Perl para este modificador.

S

Cuando un patrón va a ser usado varias veces, vale la pena dedicar más tiempo a analizarlo para acelerar el proceso de comparaciones. Si éste modificador es definido, entonces se realizará este análisis adicional. Por el momento, el estudio de un patrón es útil sólo para patrones no-anclados que no tienen un carácter de inicio único arreglado.

U (PCRE_UNGREEDY)

Este modificador invierte la "ambición" de los cuantificadores, de modo que no sean codiciosos por defecto, en su lugar se vuelven codiciosos si son seguidos por un "?". No es compatible con Perl. También puede definirse con un modificador (?U) al interior del patrón o

X (PCRE_EXTRA)

Este modificador activa funcionalidad adicional de PCRE que no es compatible con Perl. Cualquier barra invertida en un patrón que sea seguida por una letra que no tenga un significado especial provocará un error, logrando en efecto reservar éstas combinaciones para futuras ampliaciones. Por defecto, como en Perl, una barra invertida seguida por una letra sin un significado especial es tratada como un literal. No hay otras características controladas por este modificador a la fecha de hoy.

u (PCRE_UTF8)

Este modificador activa funcionalidad adicional de PCRE que no es compatible con Perl. Las cadenas de patrones son tratadas como UTF-8. Este modificador se encuentra disponible a partir de PHP 4.1.0 o versiones posteriores en Unix, y desde PHP 4.2.3 en win32.

Sintaxis de los Patrones

Sintaxis de los Patrones -- Describe la sintaxis de expresiones regulares PCRE

Descripción

La biblioteca PCRE es un conjunto de funciones que implementa comparaciones con patrones de expresiones regulares usando usando la misma sintaxis y semántica de Perl 5, con tan solo unas pocas diferencias (ver más adelante). La implementación actual corresponde a Perl 5.005.

Diferencias con Perl

Las diferencias descritas aquí existen con respecto a Perl 5.005.

1. Por defecto, un caracter de espacio en blanco es cualquier caracter que reconozca la función isspace() de la biblioteca C, aunque es posible compilar PCRE con tablas alternativas de tipos de caracteres. Normalmente, isspace() coincide con el espacio, la alimentación de página, la nueva línea, el retorno de carro, el tabulador horizontal y el tabulador vertical. Perl 5 ya no incluye el tabulador vertical en su conjunto de caracteres de espacio en blanco. La secuencia de escape \v que permaneció durante mucho tiempo en la documentación de Perl nunca fue reconocida en realidad. Sin embargo, el caracter mismo era tratado como espacio en blanco por lo menos hasta 5.002. En 5.004 y 5.005 no coincide con \s.

2. PCRE no permite cuantificadores de repetición en aserciones hacia adelante. Perl las permite, pero no quieren decir lo que probablemente piense. Por ejemplo, (?!a){3} no quiere decir que los siguientes tres caracteres no sean "a". Simplemente indica que el siguiente caracter no sea "a" tres veces.

3. Los sub-patrones de captura que aparecen al interior de aserciones negativas hacia adelante son contados, pero sus entradas en el vector de desplazamientos no son definidas. Perl define sus variables numéricas a partir de cualquiera de tales patrones que coinciden antes que la aserción falle en coincidir algo (y por lo tanto tiene éxito), pero solo si la aserción negativa hacia adelante contiene una sola rama.

4. Aunque los caracteres de cero binario son soportados en la cadena de asunto, no son permitidos en una cadena de patrón porque éstas son pasadas como un cadena normal de C, terminada en cero. La secuencia de escape "\0" puede ser usada en el patrón para representar el cero binario.

5. Las siguientes secuencias de escape de Perl no son soportadas: \l, \u, \L, \U, \E, \Q. De hecho, estas son implementadas por el mecanismo de gestión general de cadenas de Perl y no son parte de su motor de comparación de patrones.

6. La aserción \G de Perl no es soportada, ya que no es relevante para las coincidencias sencillas de patrones.

7. Obviamente, PCRE no soporta la construcción (?{código}).

8. En la actualidad hay algunas peculiaridades en Perl 5.005_02 con respecto a los grupos de cadenas capturadas cuando parte de un patrón se repite. Por ejemplo, al coincidir "aba" con el patrón /^(a(b)?)+$/ se define $2 como "b", pero al coincidir "aabbaa" con /^(aa(bb)?)+$/ deja $2 sin definir. Sin embargo, si el patrón se modifica a /^(aa(b(b))?)+$/ entonces $2 (y $3) se definen. En Perl 5.004 se define $2 en ambos casos, y también ocurre en PCRE. Si en el futuro Perl se adapta a un estilo consistente que sea diferente, PCRE puede cambiar para ajustarse.

9. Otra discrepancia aún no resuelta consiste en que, en Perl 5.005_02, el patrón /^(a)?(?(1)a|b)+$/ coincide con la cadena "a", pero en PCRE no. Sin embargo, tanto en Perl como en PCRE /^(a)?a/ coincide con "a", dejando $1 sin definir.

10. PCRE ofrece algunas extensiones a las capacidades de expresiones regulares de Perl:

    a. Aunque las aserciones hacia atrás deben coincidir con cadenas de longitud fija, cada rama alternativa de una aserción hacia atrás puede coincidir con una longitud diferente de cadena. Perl 5.005 requiere que todas ellas tengan la misma longitud.

    b. Si PCRE_DOLLAR_ENDONLY se define y PCRE_MULTILINE no, el meta-caracter $ sólo coincide al final absoluto de la cadena.

    c. Si se define PCRE_EXTRA, una barra invertida seguida de una letra sin significado especial provoca un error.

    d. Si se define PCRE_UNGREEDY, la ambición de los cuantificadores de repetición es invertida, es decir, no son ambiciosos por defecto, pero si son seguidos de un signo de interrogación, sí lo serán.

Detalles de las Expresiones Regulares

Introducción

La sintaxis y semántica de las expresiones regulares soportadas por PCRE se describe a continuación. Las expresiones reglurares son descritas en la documentación de Perl y en varios libros más, algunos de los cuales contienen numerosos ejemplos. El libro "Mastering Regular Expressions" de Jeffrey Friedl, publicado por O'Reilly (ISBN 1-56592-257-3), las cubre con gran detalle. El propósito de la presente descripción es el de servir como documentación de referencia.

Una expresión regular es un patrón que es comparado contra una cadena de asunto, de izquierda a derecha. La mayoría de caracteres se representan a ellos mismos en un patrón, y coinciden con el caracter correspondiente en el asunto. Como ejemplo trivial, el patrón *The quick brown fox* coincide con una porción de la cadena de asunto que sea idéntica al patrón dado.

Meta-caracteres

El poder de las expresiones regulares proviene de la habilidad de incluir alternativas y repeticiones en el patrón. Éstos recursos son codificados en el patrón mediante el uso de *meta-caracteres*, los cuales no se representan a ellos mismos, en su lugar, son interpretados de una forma especial.

Hay dos conjuntos diferentes de meta-caracteres: aquellos que son reconocidos en cualquier parte dentro del patrón excepto entre corchetes cuadrados, y aquellos que son reconocidos entre corchetes cuadrados. Por fuera de tales corchetes, los meta-caracteres son los siguientes:

\

caracter de escape general con varios usos

^

aserción de inicio de la cadena de asunto (o línea, en modo multilínea)

$

aserción de fin de la cadena de asunto (o línea, en modo multilínea)

.

coincide con cualquier caracter excepto la nueva línea (por defecto)

[

inicia la definición de clases de caracteres

]

fin de la definición de clases de caracteres

|

inicio de rama alternativa

(

inicio de sub-patrón

)

fin de sub-patrón

?

extiende el significado de (, también es el cuantificador 0 ó 1 también es el cuantificador de mínimo

*

cuantificador cero o más

+

cuantificador uno o más

{

cuantificador de inicio de valores mínimo/máximo

}

cuantificador de final de valores mínimo/máximo

Un segmento de un patrón que se encuentre entre corchetes cuadrados es llamado una "clase de caracteres". En una clase de caracteres, los únicos meta-caracteres son:

\

caracter general de escape

^

niega la clase, pero sólo si se trata del primer caracter

-

indica un rango de caracteres

]

finaliza la clase de caracteres

Las secciones siguientes describen el uso de cada uno de los meta-caracteres.

barra invertida

El caracter de barra invertida tiene varios usos. Primero, si es seguido por un caracter no-alfanumérico, remueve cualquier significado que el caracter pueda tener. Este uso de la barra invertida como un caracter de escape se aplica tanto dentro como fuera de las clases de caracteres.

Por ejemplo, si desea crear una coincidencia con un caracter "*", debe escribir "*" en el patrón. Esto es aplicable bien sea que el caracter siguiente hubiese sido interpretado como un meta-caracter o no, así que siempre es seguro preceder un caracter no-alfanumérico con "\" para indicar que se representa a él mismo. En particular, si desea crear una coincidencia con una barra invertida, escriba "\\".

Si un patrón es compilado con la opción PCRE_EXTENDED, los espacios en blanco del patrón (fuera de una clase de caracteres) y los caracteres entre un "#", fuera de una clase de caracteres, y el siguiente salto de línea son ignorados. Una barra invertida de escape puede ser usada para incluir un espacio en blanco o un caracter "#" como parte del patrón.

Un segundo uso de la barra invertida ofrece una forma de codificar caracteres no-imprimibles en los patrones de una forma visible. No hay restricciones sobre la apariencia de los caracteres no-imprimibles, aparte del cero binario que finaliza un patrón, pero cuando un patrón está siendo preparado mediante la edición de texto, usualmente es más fácil usar una de las siguientes secuencias de escape en lugar de los caracteres binarios que representan:

\a

alarma, esto es, el caracter BEL (hexadecimal 07)

\cx

"control-x", en donde x es cualquier caracter

\e

escape (hexadecimal 1B)

\f

alimentación de página (hexadecimal 0C)

\n

nueva línea (hexadecimal 0A)

\r

retorno de carro (hexadecimal 0D)

\t

tabulador (hexadecimal 09)

\xhh

caracter con código hexadecimal hh

\ddd

caracter con código octal ddd, o referencia hacia atrás

El efecto preciso de "*\cx*" es como sigue: si "*x*" es una letra minúscula, ésta es convertida a mayúscula. Entonces el sexto bit del caracter (40 en hexadecimal) es invertido. Por lo tanto, "*\cz*" se convierte en 1A en hexadecimal, pero "*\c{*" se convierte en 3B en hexadecimal, mientras que "*\c;*" se convierte en 7B en hexadecimal.

Después de "*\x*", son leidos hasta dos dígitos hexadecimales (las letras pueden ser mayúsculas o minúsculas).

Después de "*\0*", son leídos hasta dos dígitos octales más. En ambos casos, si hay menos de dos dígitos, se usarán sólo los que estén presentes. Por lo tanto, la secuencia "*\0\x\07*" especifica dos ceros binarios seguidos de un caracter BEL. Asegúrese de indicar dos dígitos después del cero inicial si el caracter que sigue es en sí un dígito octal.

La gestión de una barra invertida seguida por un dígito diferente de cero es complicada. Por fuera de una clase de caracteres, PCRE lee el dígito y cualquier otro que le siga como un número decimal. Si el número es menor que diez, o si han habido al menos tantos paréntesis izquierdos de captura en la expresión, entonces la secuencia entera es tomada como una *referencia hacia atrás*. Una descripción de cómo trabaja esto es presentada más adelante, tras la discusión sobre sub-patrones con paréntesis.

Al interior de una clase de caracteres, o si el número decimal es mayor que 9 y no han habido tantos sub-patrones de captura, PCRE lee de nuevo hasta tres dígitos octales que sigan a la barra invertida, y genera un byte sencillo a partir de los ocho bits menos significativos del valor. Cualquier dígito subsiguiente se representa a él mismo. Por ejemplo:

\040

es otro modo de escribir un espacio

\40

es lo mismo, siempre que haya menos de cuarenta sub-patrones de captura previos

\7

siempre es una referencia hacia atrás

\11

puede ser una referencia hacia atrás, u otra forma de escribir un tabulador

\011

siempre es un tabulador

\0113

es un tabulador seguido del caracter "3"

\113

es el caracter con el código octal 113 (ya que no puede haber más de 99 referencias hacia atrás)

\377

es un byte que consiste completamente de bits 1

\81

puede ser una referencia hacia atrás, o un cero binario seguido por los caracteres "8" y "1"

Note que los valores octales del 100 o números más grandes no deben iniciar con un cero, ya que no se leen más de tres dígitos octales.

Todas las secuencias que definen el valor de un byte sencillo pueden ser usadas tanto dentro como fuera de las clases de caracteres. Adicionalmente, la secuencia "*\b*" es interpretada como el caracter backspace (hexadecimal 08) al interior de una clase de caracteres. Por fuera de una clase de caracteres tiene un significado diferente (ver más adelante).

El tercer uso de la barra invertida es para especificar tipos genéricos de caracteres:

\d

cualquier dígito decimal

\D

cualquier caracter que no sea un dígito decimal

\s

cualquier caracter de espacio en blanco

\S

cualquier caracter que no sea un espacio en blanco

\w

cualquier caracter de "palabra"

\W

cualquier caracter que no sea de "palabra"

Cada pareja de las secuencias de escape divide el conjunto global de caracteres en dos grupos separados. Cualquier caracter dado coincide con uno, y sólo uno, de cada pareja.

Un caracter de "palabra" es cualquier letra o dígito, o el caracter de subrayado, esto quiere decir, cualquier caracter que pueda ser parte de una "*palabra*" en Perl. La definición de letras y dígitos es controlada por las tablas de caracteres de PCRE, y puede variar si se están efectuando coincidencias específicas a localidades (vea "Soporte de localidades" más arriba). Por ejemplo, en la localidad "fr" (Francia), algunos códigos de caracteres mayores a 128 son usados para letras con acentos, y éstas coinciden con *\w*.

Estas secuencias de tipos de caracter pueden aparecer tanto dentro como fuera de las clases de caracteres. Cada una coincide con un caracter del tipo apropiado. Si el punto de coincidencia actual es el final de la cadena de asunto, todas las secuencias fallan, ya que no hay caracteres a coincidir.

El cuarto uso de la barra invertida es para ciertas aserciones simples. Una aserción especifica una condición que tiene que cumplirse en un punto particular de una coincidencia, sin consumir caracter alguno de la cadena de asunto. El uso de sub-patrones para aserciones más complicadas se describe más adelante. Las aserciones de barra invertida son

\b

límite de palabra

\B

no-límite de palabra

\A

inicio de la cadena de asunto (independiente del modo multilínea)

\Z

fin de la cadena de asunto o una nueva línea al final (independiente del modo multilínea)

\z

fin de la cadena de asunto (independiente del modo multilínea)

Estas aserciones no pueden aparecer dentro de clases de caracteres (pero note que "*\b*" tiene un significado diferente, el cual es el caracter backspace, dentro de una clase de caracteres).

Un límite de palabra es una posición en la cadena de asunto en donde el caracter actual y el anterior no coinciden ambos con *\w* o *\W* (es decir, uno coincide con *\w* y el otro coincide con *\W*), o se puede tratar del principio o el final de la cadena, si el primer o último caracter coincide con *\w*, respectivamente.

Las aserciones *\A*, *\Z*, y *\z* se diferencian de los caracteres tradicionales circunflejo y dólar (descritos más adelante) en que las primeras sólo coinciden al inicio y final absolutos de la cadena de asunto, independientemente de las opciones definidas. No son influenciadas por las opciones PCRE_MULTILINE o PCRE_DOLLAR_ENDONLY. La diferencia entre *\Z* y *\z* es que *\Z* coincide antes de una nueva línea que sea el último caracter de la cadena como también al final de la cadena, mientas que *\z* sólo coincide al final.

El circunflejo y el dólar

Por fuera de una clase de caracteres, en el modo predeterminado de coincidencia, el caracter circunflejo es una aserción que sólo es verdadera si el punto de coincidencia actual es el inicio de la cadena de asunto. Al interior de una clase de caracteres, el circunflejo tiene un significado completamente distinto (ver más adelante).

El circunflejo no necesita ser el primer caracter del patrón si se involucra un número de alternativas, pero debe ser la primer cosa en cada alternativa en la que aparezca si se espera que el patrón coincida con esa rama. Si todas las alternativas posibles empiezan con un circunflejo, esto es, si el patrón está limitado a coincidir sólo con en el inicio del asunto, se dice que es un patrón "anclado". (También hay otras construcciones que pueden hacer que un patrón sea anclado.)

Un caracter de dólar es una aserción que es verdadera sólo si el punto de coincidencia actual se encuentra al final de la cadena de asunto, o inmediatamente antes de un caracter de nueva línea que sea el último caracter en la cadena (por defecto). El dólar no necesita ser el último caracter del patrón si hay varias alternativas involucradas, pero debe ser el último elemento en cada rama en la que aparezca. El dólar no tiene un significado especial en una clase de caracteres.

El significado del dólar puede ser modificado para que coincida sólo al final absoluto de la cadena, definiendo la opción PCRE_DOLLAR_ENDONLY en tiempo de compilación o a la hora de efectuar la comparación. Esto no afecta a la aserción \Z.

Los significados de los caracteres circunflejo y dólar son modificados si la opción PCRE_MULTILINE es definida. Cuando éste es el caso, éstos caracteres coinciden inmediatamente antes e inmediatamente después de un caracter "\n" interno, respectivamente, además de coincidir con el inicio y el final de la cadena de asunto. Por ejemplo, el patrón /^abc$/ coincide con la cadena de asunto "def\nabc" en modo multilínea, pero no en otro caso. Consecuentemente, los patrones que son anclados en modo de línea sencilla ya que todas las ramas empiezan con "^" no son anclados en modo multilínea. La opción PCRE_DOLLAR_ENDONLY es ignorada si PCRE_MULTILINE es definido.

Tenga en cuenta que las secuencias \A, \Z y \z pueden ser usadas para coincidir con el inicio y el final del asunto en ambos modos, y si todas las ramas de un patrón comienzan con \A, el patrón siempre es anclado, independientemente de si PCRE_MULTILINE es definido o no.

PUNTO

Por fuera de una clase de caracteres, un punto en el patrón coincide con cualquier caracter del asunto, incluyendo caracteres no-imprimibles, pero no el salto de línea (por defecto). Si la opción PCRE_DOTALL es definida, entonces los puntos coinciden con los saltos de línea también. El manejo del punto es completamente independiente del uso del circunflejo y el dólar, dado que la

única relación entre ellos es que ambos casos involucran caracteres de nueva línea. El punto no tiene un significado especial dentro de una clase de caracteres.

Corchetes cuadrados

Un corchete cuadrado de apertura inicia una clase de caracteres, terminada por un corchete cuadrado de cierre. Un corchete cuadrado de cierre por sí solo no es especial. Si un corchete cuadrado de cierre es requerido como un miembro de la clase, debería ser el primer caracter de datos en la clase (después de un circunflejo inicial, si está presente) o escapado con una barra invertida.

Una clase de caracteres coincide con un caracter único en el asunto; el caracter debe estar en el conjunto de los caracteres definidos por la clase, a menos que el primer caracter en la clase sea un circunflejo, en cuyo caso el caracter del asunto no debe estar en el conjunto definido por la clase. Si un circunflejo es necesitado realmente como un miembro de la clase, asegúrese de que no sea el primer caracter, o escápelo con una barra invertida.

Por ejemplo, la clase de caracteres [aeiou] coincide con cualquier vocal minúscula, mientras que [^aeiou] coincide con cualquier caracter que no sea una vocal minúscula. Note que un circunflejo es una notación conveniente para especificar los caracteres que están en la clase enumerando aquellos que no lo están. No es una aserción: aun consume un caracter de la cadena de asunto, y falla si el apuntador actual está al final de la cadena.

Cuando se recurre a las comparaciones insensibles a mayúsculas y minúsculas, cualquier letra en una clase representa ambas versiones, por ejemplo, un patrón insensible a mayúsculas y minúsculas [aeiou] coincide tanto con "A" como con "a", y un patrón insensible a mayúsculas y minúsculas [^aeiou] no coincide con "A", mientras que una versión sensible lo haría.

El caracter de nueva línea nunca es tratado de un modo especial entra las clases de caracteres, independientemente de los valores de las opciones PCRE_DOTALL o PCRE_MULTILINE. Una clase como [^a] siempre coincidirá con una nueva línea.

El caracter menos (guión) puede ser usado para especificar un rango de caracteres en una clase de caracteres. Por ejemplo, [d-m] coincide con cualquier letra entre d y m, ambas inclusive. Si un caracter menos es requerido en una clase, debe ser escapado con una barra invertida, o aparecer en una posición en donde no pueda ser interpretado como indicador de rango, normalmente como primer o último caracter de la clase.

No es posible tener el caracter literal "]" como el caracter final de un rango. Un patrón como [W-]46] es interpretado como una clase de dos caracteres ("W" y "-") seguida por la cadena literal "46]", así que coincidiría con "W46]" o "-46]". Sin embargo, si el caracter "]" es escapado con una barra invertida, éste es interpretado como el final del rango, así que [W-\]46] es interpretado como una clase única que contiene un rango seguido por dos caracteres diferentes. La representación octal o hexadecimal de "]" puede ser usada también para finalizar un rango.

Los rangos trabajan en el orden de la secuencia ASCII. Pueden ser usados también para caracteres especificados numéricamente, por ejemplo [\000-\037]. Si un rango que incluye letras es usado cuando es definida la comparación insensible a mayúsculas y minúsculas, el rango coincide las letras en cualquiera de los casos. Por ejemplo, [W-c] es ñalente a [][\^_`wxyzabc], efectuando la coincidencia insensible a mayúsculas y minúsculas, y si las tablas de caracteres para la localidad "fr" están en uso, entonces [\xc8-\xcb] coincide con los caracteres E acentuados en ambos casos.

Los tipos de caracteres \d, \D, \s, \S, \w, y \W también pueden aparecer en una clase de caracteres, y añaden los caracteres que ellos representan a la clase. Por ejemplo, [\dABCDEF] coincide con cualquier dígito hexadecimal. Un circunflejo puede ser convenientemente usado con los tipos de

caracter en mayúscula para especificar un conjunto más restringido de caracteres que el de una comparación con tipo en minúscula. Por ejemplo, la clase [^\W_] coincide con cualquier letra o dígito, pero no con el signo de subrayado.

Todos los caracteres no-alfanuméricos diferentes a \, -, ^ (al comienzo) y el caracter] de cierre no tienen un significado especial en una clase de caracteres, pero no hace daño que se encuentren escapados..

Barra vertical

Los caracteres de barra vertical son usados para separar patrones alternativos. Por ejemplo, el patrón *gilbert|sullivan* coincide o bien con "gilbert" o con "sullivan". Puede usarse cualquier número de alternativas, y se permiten alternativas vacías (que coinciden con la cadena vacía). El proceso de comparación prueba con cada alternativa de izquierda a derecha, y la primera que tenga éxito es usada. Si las alternativas están al interior de un sub-patrón (definido más adelante), el "éxito" quiere decir que coincida con el resto del patrón principal como también con la alternativa en el sub-patrón.

Definición de opciones internas

Los valores de PCRE_CASELESS, PCRE_MULTILINE, PCRE_DOTALL, PCRE_UNGREEDY, y PCRE_EXTENDED pueden ser modificados desde el interior del patrón por una secuencia de letras de opciones de Perl encerradas entre "(?" y ")". Las letras de opciones son

Tabla 1. Letras de opciones internas

i	para PCRE_CASELESS
m	para PCRE_MULTILINE
s	para PCRE_DOTALL
x	para PCRE_EXTENDED
U	para PCRE_UNGREEDY

Por ejemplo, (?im) define una comparación insensible a mayúsculas y minúsculas y en modo multilínea. También es posible eliminar éstas opciones precediendo las letras con un guión, así como se permite también una combinación de activaciones y desactivaciones como (?im-sx), la cual define PCRE_CASELESS y PCRE_MULTILINE al mismo tiempo que desactiva PCRE_DOTALL y PCRE_EXTENDED. Si una letra aparece antes y después del guión, la opción será desactivada.

El contexto de éstas opciones cambia dependiendo del lugar en el que ocurra la definición. Para las definiciones que son hechas por fuera de sub-patrones (definidos más adelante), el efecto es el mismo que si la opción fuera activada o desactivada al inicio de la coincidencia. Los siguientes patrones se comportan todos de la misma manera:

(?i)abc
a(?i)bc
ab(?i)c
abc(?i)

que a su vez tienen el mismo efecto que compilar el patrón abc con la opción PCRE_CASELESS activa. En otras palabras, tales definiciones de "nivel superior" se aplican en todo el patrón (a menos

que haya otros cambios al interior de sub-patrones). Si hay más de una definición de la misma opción en el nivel superior, la definición más a la derecha es usada.

Si un cambio de opción sucede dentro de un sub-patrón, el efecto es diferente. Este es un cambio respecto a la conducta de Perl 5.005. Un cambio de opción dentro de un sub-patrón afecta sólo a la parte del sub-patrón que lo sigue, de modo que *(a(?i)b)c* coincide con abc y aBc y ninguna otra cadena (asumiendo que no se está usando PCRE_CASELESS). De esta forma, las opciones pueden definirse para tener diferentes significados en diferente partes del patrón. Cualquier cambio realizado en una alternativa ciertamente se aplica a ramas subsecuentes al interior del mismo sub-patrón. Por ejemplo, *(a(?i)b|c)* coincide con "ab", "aB", "c", y "C", aun cuando al coincidir con "C", la primera rama es abandonada antes de definir la opción. Esto es porque los efectos de definir de opciones ocurren en tiempo de compilación. De otro modo, ocurriría un comportamiento muy extraño.

Las opciones específicas de PCRE PCRE_UNGREEDY y PCRE_EXTRA pueden ser modificadas del mismo modo que las opciones compatibles con Perl usando los caracteres U y X respectivamente. La opción bandera (?X) es especial en el sentido en que siempre debe ocurrir antes que cualquier otra característica adicional que active en el patrón, incluso cuando es definida en el nivel superior. Su mejor ubicación es el inicio.

sub-patrones

Los sub-patrones son delimitados por paréntesis, y pueden estar anidados. Marcar parte de un patrón como un sub-patrón logra dos cosas:

1. Ubica un conjunto de alternativas. Por ejemplo, el patrón *cat(aract|erpillar|)* coincide con una de las palabras "cat", "cataract", o "caterpillar". Sin los paréntesis, coincidiría con "cataract", "erpillar" o la cadena vacía.

2. Define el sub-patrón como un sub-patrón de captura (como se definió anteriormente). Cuando el patrón completo coincida, esa porción de la cadena de asunto que coincidió con el sub-patrón es devuelta al origen mediante el argumento *ovector* de **pcre_exec()**. Los paréntesis de apertura son contados de izquierda a derecha (empezando desde 1) para obtener los números de los sub-patrones de captura.

Por ejemplo, si la cadena "the red king" es comparada contra el patrón *the ((red|white) (king|queen))* las sub-cadenas capturadas son "red king", "red", y "king", y son numeradas como 1, 2 y 3.

El hecho de que los simples paréntesis realicen dos funciones no siempre es útil. Con frecuencia se presenta el caso en el que un sub-patrón de agrupamiento es requerido sin necesidad de una captura. Si un paréntesis de apertura es seguido por "?:", el sub-patrón no realiza ninguna captura, y no es contado cuando se compute el número de sub-patrones subsiguientes capturados. Por ejemplo, si la cadena "the white queen" es comparada con el patrón *the ((?:red|white) (king|queen))* las sub-cadenas capturadas son "white queen" y "queen", y son numeradas como 1 y 2. El número máximo de sub-cadenas capturadas es de 99, y el número máximo de todos los sub-patrones, de captura o no, es de 200.

Como un atajo conveniente, si cualquiera de las opciones se requiere al inicio de un sub-patrón que no sea de captura, las letras de las opciones pueden aparecer entre los caracteres "?" y ":". Por lo tanto, los dos patrones

```
(?i:saturday|sunday)
(?:(?i)saturday|sunday)
```

coinciden con exactamente el mismo conjunto de cadenas. Dado que las ramas alternativas son probadas de izquierda a derecha, y las opciones no son reestablecidas hasta el final del sub-patrón, una definición de opción en una rama afecta las ramas subsecuentes, así que los patrones anteriores coinciden con "SUNDAY", al igual que con "Saturday".

Repetición

La repetición es especificada por cuantificadores, los cuales pueden ir tras cualquiera de los siguientes elementos:

- un caracter sencillo, posiblemente escapado
- el meta-caracter .
- una clase de caracteres
- una referencia hacia atrás (vea la siguiente sección)
- un sub-patrón entre paréntesis (a menos que se trate de una aserción - vea más adelante)

El cuantificador general de repetición indica un número mínimo y un número máximo de coincidencias permitidas, dando los dos números entre corchetes ondulados (llaves), separados por una coma. Los números deben ser menores a 65536, y el primero debe ser menor o igual al segundo. Por ejemplo: *z{2,4}* coincide con "zz", "zzz", o "zzzz". Una llave de cierre por sí sola no es un caracter especial. Si el segundo número es omitido, pero aparece la coma, entonces no hay límite superior; si el segundo número y la coma son omitidos ambos, el cuantificador indica el número exacto de repeticiones requeridas. Por lo tanto *[aeiou]{3,}* coincide con al menos 3 vocales sucesivas, pero podría coincidir con muchas más, mientras que *\d{8}* coincide con exactamente ocho dígitos. Una llave de apertura que aparezca en una posición en donde no se permite un cuantificador, o una que no coincida con la sintaxis de un cuantificador, es tomada como un caracter literal. Por ejemplo, {,6} no es un cuantificador, sino una cadena literal de cuatro caracteres.

Se permite el uso del cuantificador {0}, lo que provoca que la expresión se comporte como si el elemento anterior y el cuantificador no estuvieran presentes.

Por conveniencia (y compatibilidad histórica) los tres cuantificadores más comunes tienen abreviaciones de un solo caracter:

Tabla 2. Cuantificadores de caracter-único

*	ñalente a *{0,}*
+	ñalente a *{1,}*
?	ñalente a *{0,1}*

Es posible construir ciclos infinitos mediante un sub-patrón que no pueda coincidar con ningún caracter con un cuantificador que no tenga límite superior, por ejemplo: *(a?)**

Las primeras versiones de Perl y PCRE solían producir un error en tiempo de compilación para tales patrones. Sin embargo, dado que existen casos en donde esto puede ser útil, tales patrones son aceptados ahora, pero si cualquier repetición del sub-patrón no coincide realmente con ningún caracter, el ciclo es interrumpido a la fuerza.

Por defecto, los cuantificadores son "ambiciosos", lo que quiere decir, coinciden con tanto material como les es posible (hasta el número máximo de veces permitido), sin provocar que el resto del patrón falle. El ejemplo clásico en el que esto causa problema es a la hora de crear coincidencias con comentarios en programas en C. Éstos aparecen entre las secuencias /* y */ y, al interior de la secuencia, los caracteres * y / pueden aparecer individualmente. Un intento por coincidir comentarios en C al aplicar el patrón */*.**/* sobre la cadena */* primer comentario */ no comentado /* segundo comentario */* falla, ya que coincide con la cadena entera debido a la ambición del elemento .*

Sin embargo, si un cuantificador es seguido por un signo de interrogación, entonces deja de ser ambicioso, y en su lugar coincide el mínimo número de veces posibles, de tal suerte que el patrón */*.*?*/* hace lo correcto con los comentarios en C. El significado de los varios cuantificadores no se modifica en otro modo, tan sólo el número preferido de coincidencias. No confunda éste uso del signo de interrogación con su uso como un cuantificador por sí solo. Debido a que tiene dos usos, a veces puede aparecer dos veces seguidas, como en *\d??\d* caso que coincide con un dígito de ser posible, pero puede coincidir con dos si ese el único modo en que el resto del patrón coincida.

Si se encuentra definida la opción PCRE_UNGREEDY (la cual no está disponible en Perl) entonces los cuantificadores no son ambiciosos por defecto, pero cada uno por separado puede serlo cuando a continuación de ellos se encuentra un signo de interrogación. En otras palabras, invierte la conducta predeterminada.

Cuando un sub-patrón entre paréntesis es cuantificado con un número mínimo de repeticiones superior a 1 o con un límite máximo, se necesita mayor almacenamiento para el patrón compilado, en proporción al tamaño del mínimo o del máximo.

Si un patrón empieza con .* o .{0,} y la opción PCRE_DOTALL (ñalente a /s en Perl) es definida, permitiendo de esa forma que . coincida con nuevas líneas, entonces el patrón es anclado implícitamente, ya que cualquier cosa a continuación será comparada contra cada posición de caracter en la cadena de asunto, así que no hay razones para reintentar la coincidencia en su totalidad en cualquier posición luego de la primera. PCRE trata tales patrones como si estuvieran precedidos por \A. En los casos donde se conoce que la cadena de asunto no contiene nuevas líneas, vale la pena definir PCRE_DOTALL cuando el patrón comienza con .* para obtener esta optimización, o alternativamente usar ^ para indicar el anclamiento explícitamente.

Cuando un sub-patrón de captura es repetido, el valor capturado es la sub-cadena que coincidió con la iteración final. Por ejemplo, luego de que *(tweedle[dume]{3}\s*)+* ha coincidido con "tweedledum tweedledee" el valor de la sub-cadena capturada es "tweedledee". Sin embargo, si hay sub-patrones de captura anidados, los valores capturados correspondientes pueden haber sido definidos en las iteraciones anteriores. Por ejemplo, después de que */(a|(b))+/* coincide con "aba", el valor de la segunda sub-cadena capturada es "b".

REFERENCIAS HACIA ATRÃƒÂ S

Por fuera de una clase de caracteres, una barra invertida seguida por un digito mayor que cero (y posiblemente más dígitos) es una referencia hacia atrás a un sub-patrón de captura anterior (es decir, a su izquierda) en el patrón, siempre y cuando existan tantos paréntesis izquierdos de captura.

Sin embargo, si el número decimal a continuación de la barra invertida es menor que diez, siempre es tomado como una referencia hacia atrás, y causa un error sólo si no hay los suficientes paréntesis izquierdos de captura en todo el patrón. En otras palabras, los paréntesis que son referidos no necesitan estar a la izquierda de la referencia para números menores que diez. Vea la sección anterior titulada "Barra invertida" para más detalles sobre el manejo de los dígitos que siguen a una barra invertida.

Una referencia hacia atrás coincide con cualquier cosa que haya coincidido realmente con el sub-patrón de captura en la cadena de asunto actual, en lugar de hacerlo con cualquier cosa que coincida con el sub-patrón mismo. De modo que el patrón *(sens|respons)e and \1ibility* coincide con "sense and sensibility" y "response and responsibility", pero no "sense and responsibility". Si se está aplicando una comparación sensible a mayúsculas y minúsculas al momento de la referencia hacia atrás, entonces la distinción de las letras es importante. Por ejemplo, *((?i)rah)\s+\1* coincide con "rah rah" y "RAH RAH", pero no "RAH rah", incluso cuando el sub-patrón de captura original fue comparado de forma insensible a mayúsculas y minúsculas.

Puede haber más de una referencia hacia atrás hacia el mismo sub-patrón. Si un sub-patrón no ha sido usado realmente en una coincidencia particular, entonces cualquier referencia hacia atrás hacia aquél siempre falla. Por ejemplo, el patrón *(a|(bc))\2* siempre falla si comienza coincidiendo con "a" en lugar de "bc". Ya que puede haber hasta 99 referencias hacia atrás, todos los dígitos que siguen a la barra invertida son tomados como parte de un potencial número de referencia hacia atrás. Si el patrón continúa con un caracter de dígito, entonces debe ser usado algún delimitador para terminar la referencia hacia atrás. Si la opción PCRE_EXTENDED es definida, este puede ser el espacio en blanco. De otro modo, un comentario vació puede ser usado.

Una referencia hacia atrás que ocurra dentro del paréntesis al cual hace referencia falla cuando el sub-patrón es usado por primera vez, así que, por ejemplo, (a\1) nunca crea coincidencias. Sin embargo, tales referencia pueden ser útiles al interior de sub-patrones repetidos. Por ejemplo, el patrón *(a|b\1)+* coincide con cualquier número de "a"s y también con "aba", "ababaa" etc. Para cada iteración del sub-patrón, la referencia hacia atrás coincide con la cadena de caracteres correspondiente a la iteración anterior. Para que esto funcione, el patrón debe ser tal que la primera iteración no necesite coincidir con la referencia hacia atrás. Esto puede lograrse usando alternaciones, como en el ejemplo anterior, o por medio de un cuantificador con un mínimo de cero.

Aserciones

Una aserción es una prueba sobre los caracteres a continuación o antes del punto actual de coincidencia que no consume caracteres en realidad. Las aserciones simples codificadas como \b, \B, \A, \Z, \z, ^ y $ son descritas anteriormente. Las aserciones más complicadas son codificadas como sub-patrones. Hay dos tipos: aquellas que trabajan con material más adelante de la posición actual en la cadena de asunto y aquellas que lo hacen con material hacia atrás.

Un sub-patrón de aserción es comparado del modo usual, excepto que no causa que el punto actual de coincidencia cambie. Las aserciones hacia adelante comienzan con (?= en el caso de aserciones positivas y (?! para las negativas. Por ejemplo, *\w+(?=;)* coincide con una palabra seguida por un punto-y-coma. pero no incluye el punto-y-coma en la coincidencia, y *foo(?!bar)* coincide con cualquier ocurrencia de "foo" que no sea seguida por "bar". Note que el patrón, en apariencia semejante, *(?!foo)bar* no encuentra una ocurrencia de "bar" que sea precedida por algo diferente de "foo"; encuentra cualquier ocurrencia de "bar", ya que la aserción (?!foo) es siempre verdadera cuando los siguientes tres caracteres son "bar". Una aserción hacia atrás es necesaria para conseguir este efecto.

Las aserciones hacia atrás comienzan con (?<= para las aserciones positivas y (?<! para las negativas. Por ejemplo, *(?<!foo)bar* encuentra una ocurrencia de "bar" que no es precedida por "foo". Los contenidos de una aserción hacia atrás son restringidos de tal forma que todas las cadenas con las que coinciden deben tener una longitud fija. Sin embargo, si hay varias alternativas, no todas tienen que tener la misma longitud. Por lo tanto *(?<=bullock|donkey)* se permite, pero *(?<!dogs?|cats?)* genera un error en tiempo de compilación. Las ramas que coinciden con cadenas de diferentes longitudes son permitidas sólo en el nivel superior de la aserción hacia atrás. Ésta es una extensión en comparación con Perl 5.005, en donde se requiere que todas las ramas coincidan con la misma longitud de cadena. Una aserción como *(?<=ab(c|de))* no es permitida, ya que su rama única

de nivel superior puede coincidir con dos longitudes diferentes, pero es aceptable si se reescribe para usar dos ramas de nivel superior: *(?<=abc|abde)* La implementación de las aserciones hacia atrás consiste en, para cada alternativa, mover temporalmente la posición actual hacia atrás en el ancho fijo e intentar la coincidencia. Si no hay suficientes caracteres antes de la posición actual, la coincidencia está destinada a fallar. Las aserciones hacia atrás, en unión con los sub-patrones de una sola aplicación, pueden ser particularmente útiles para las coincidencias al final de cadenas; un ejemplo es dado al final de la sección sobre sub-patrones de una aplicación.

Varias aserciones (de cualquier tipo) pueden ocurrir en sucesión. Por ejemplo, *(?<=\d{3})(?<!999) foo* coincide con "foo" precedido de tres dígitos que no sean "999". Note que cada una de las aserciones es aplicada independientemente en el mismo punto en la cadena de asunto. Primero hay un chequeo para que los tres caracteres previos sean todos dígitos, luego hay un chequeo para que los mismos caracteres no sean "999". Este patrón no coincide con "foo" precedido de seis caracteres, en donde los primeros son dígitos y los últimos tres no son "999". Por ejemplo, no coincide con "123abcfoo". Un patrón para conseguir eso es *(?<=\d{3}...)(?<!999)foo*

En este caso la primera aserción revisa los seis caracteres anteriores, y chequea que los tres primeros sean dígitos, y luego la segunda aserción chequea que los tres caracteres anteriores no sean "999".

Las aserciones puede ser anidadas en cualquier combinación. Por ejemplo, *(?<=(?<!foo)bar)baz* coincide con una ocurrencia de "baz" que sea precedida por "bar", la cual a su vez no sea precedida por "foo", mientras que *(?<=\d{3}...(?<!999))foo* es otro patrón que coincide con "foo" precedido por tres dígitos y tres caracteres cualquiera que no sean "999".

Los sub-patrones de aserción no son sub-patrones de captura, y no pueden ser repetidos, ya que no tiene sentido afirmar la misma cosa varias veces. Si una aserción de cualquier tipo contiene sub-patrones de captura en su interior, éstos son contados con el propósito de numerar los sub-patrones de captura en todo el patrón. Sin embargo, la captura de subcadenas solo se lleva a cabo en las aserciones positivas, porque no tiene sentido para las negativas.

Las aserciones cuentan para el máximo de 200 sub-patrones entre paréntesis.

Sub-patrones de una sola aplicación

Tanto con las repeticiones máximas como en las mínimas, el hecho de que falle de lo que se encuentra a continuación causa por lo general que el item repetido sea re-evaluado para ver si un número diferente de repeticiones permite que el resto del patrón coincida. A veces es útil prevenir esto, ya sea cambiando la naturaleza de la coincidencia, o causando que falle antes de cuando ocurriría de otra forma, cuando el creador del patrón sabe que no tiene sentido continuar.

Considere, por ejemplo, el patrón \d+foo cuando se aplica a la línea de asunto *123456bar*

Después de coincidir con los seis dígitos y fallar al comparar con "foo", la acción normal del motor es intentar otra vez con sólo cinco dígitos para coincidir con \d+, y luego con cuatro, y así sucesivamente, antes de fallar por completo. Los sub-patrones de una aplicación ofrecen el medio de especificar que una vez una porción del patrón ha coincidido, no debe ser re-evaluada en esta manera, así que el motor se rendiría inmediamente al fallar su intento por coincidir con "foo" la primera vez. La notación es otra forma especial de paréntesis, iniciado con (?> como en este ejemplo: *(?>\d+)bar*

Este tipo de paréntesis "bloquea" la parte del patrón que contiene una vez ha coincidido, y se previene que un fallo más al interior del patrón retroceda al punto original. El retroceso hacia elementos previos funciona normalmente, después de todo.

Una descripción alternativa es que un sub-patrón de este tipo coincide con la cadena de carecteres que un patrón independiente idéntico coincidiría, si estuviera anclado en el punto actual de la cadena de asunto.

Los sub-patrones de una sola aplicación no son sub-patrones de captura. Los casos simples como el ejemplo anterior pueden verse como una repetición máxima que debe tragar todo lo que pueda. Así que, mientras que tanto \d+ como \d? están preparados para ajustar el número de dígitos con los que coinciden para hacer que el resto del patrón coincida, (?>\d+) puede coincidir sólo con un secuencia enteramente de dígitos.

Esta construcción puede, por supuesto, contener sub-patrones arbitrariamente complicados, y pueden estar anidados.

Los sub-patrones de una aplicación pueden ser usados en unión con aserciones hacia atrás para especificar coincidencias eficientes al final de la cadena de asunto. Considere un patrón sencillo como *abcd$* cuando se aplica a una cadena larga con la cual no coincide. Dado que la comparación se realiza de izquierda a derecha, PCRE buscará cada "a" en el asunto y luego verá si lo que sigue coincide con el resto del patrón. Si el patrón se especifica como ^.*abcd$ entonces el segmento .* inicial coincide con la cadena entera primero, pero cuando esto falle (ya que no habrá una "a" a continuación), retrocede para coincidir con todo menos el último caracter, luego con todo excepto los dos últimos y así sucesivamente. Una vez más, la búsqueda de "a" cubre la cadena completa, de derecha a izquierda, así que no hemos mejorado. Sin embargo, si el patrón se escribiese como ^ *(?>.*)(?<=abcd)* entonces no hay retroceso para el elemento .*; sólo puede coincidir con la cadena entera. La aserción hacia atrás subsiguiente realiza una prueba única sobre los últimos cuatro caracteres. Si falla, la coincidencia falla inmediatamente. Para cadena largas, este enfoque representa una diferencia significativa en el tiempo de procesamiento.

Cuando un patrón contiene una repetición ilimitada al interior de un sub-patrón que puede ser por sí mismo repetido un número ilimitado de veces, el uso de un sub-patrón de una aplicación es la única forma de evitar algunas coincidencias fallidas que consumen ciertamente un tiempo muy largo. El patrón *(\D+|<\d+>)*[!?]* coincide con un número ilimitado de subcadenas que consisten ya sea de no-dígitos, o dígitos entre <>, seguidos por ! o ?. Cuando coincida, se ejecuta rápido. Sin embargo, si se aplica sobre *aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa* toma un largo tiempo antes de reportar el fallo. Esto es porque la cadena puede ser dividida entre las dos repeticiones en un gran número de formas, y todas deben ser probadas. (El ejemplo usó [!?] en lugar de un caracter sencillo al final, ya que tanto PCRE como Perl cuentan con una optimización que permite reportar fallos rápidamente cuando un caracter sencillo es usado. Recuerdan el último caracter simple que es requerido para una coincidencia, y falla tempranamente si no está presente en la cadena.) Si el patrón es modificado a *((?>\D+)|<\d+>)*[!?]* las secuencias de no-dígitos no pueden ser interrumpidas, y la falla ocurre rápidamente.

Sub-patrones condicionales

Es posible hacer que el proceso de comparación obedezca a un sub-patrón condicionalmente o que elija entre dos sub-patrones alternativos, dependiendo del resultado de una aserción, o de si un sub-patrón de captura previo coincidió o no. Las dos formas posibles de sub-patrones condicionales son:

(?(condición)patrón-si)
(?(condición)patrón-si|patrón-no)

Si la condición es satisfecha, el patrón-si es usado; de otra forma el patrón-no es usado (si está presente). Si hay más de dos alternativas en el sub-patrón, se produce un error en tiempo de

compilación.

Hay dos clases de condición. Si el texto entre los paréntesis consiste de una secuencia de dígitos, entonces la condición es satisfecha si el sub-patrón de captura de ese número ha sido coincidido previamente. Consideremos el siguiente patrón, el cual contiene espacios en blanco sin significado para hacerlo más legible (asumiendo la opción PCRE_EXTENDED) y lo dividimos en tres partes para facilitar su discusión: *(\( )? [^()]+ (?(1) \))*

La primera parte coincide con un paréntesis de apertura opcional, y si ese caracter está presente, lo define como la primera subcadena capturada. La segunda parte coincide con uno o más caracteres que no sean paréntesis. La tercera parte es un sub-patrón condicional que examina si el primer conjunto de paréntesis coincidió o no. Si lo hizo, es decir, si el asunto comenzó con un paréntesis de apertura, la condición es cierta, así que el patrón-si es ejecutado y un paréntesis de cierre es requerido. De otro modo, ya que no existe un patrón-no, el sub-patrón coincide con nada. En otras palabras, este patrón coincide con una secuencia de no-paréntesis, opcionalmente entre paréntesis.

Si la condición no es una secuencia de dígitos, debe ser una aserción. Ésta puede ser una aserción positiva o negativa hacia adelante o hacia atrás. Considere este patrón, el cual contiene una vez más espacios en blanco sin significado, y con las dos alternativas en la siguiente línea:

```
(?(?=[^a-z]*[a-z])
\d{2}-[a-z]{3}-\d{2}  |  \d{2}-\d{2}-\d{2} )
```

La condición es una aserción positiva hacia adelante que coincide con una secuencia opcional de no-letras seguida por una letra. En otras palabras, examina la presencia de al menos una letra en el asunto. Si se encuentra una letra, el asunto es comparado contra la primera alternativa; de otra forma lo es con la segunda. Este patrón coincide con cadenas en una de las dos formas dd-aaa-dd o dd-dd-dd, en donde aaa son letras y dd son dígitos.

Comentarios

La secuencia (?# marca el inicio de un comentario el cual continúa hasta el siguiente paréntesis de cierre. Los paréntesis anidados no son permitidos. Los caracteres que forman un comentario no hacen parte del patrón de coincidencia en ningún caso.

Si la opción PCRE_EXTENDED es definida, un caracter # no escapado por fuera de una clase de caracteres crea un comentario que continúa hasta el próximo caracter de nueva línea en el patrón.

Patrones recursivos

Considere el problema de coincidir con una cadena entre paréntesis, permitiendo un número ilimitado de paréntesis anidados. Sin el uso de recursiones, lo mejor que puede lograrse es usar un patrón que coincida hasta un número límite de profundidad en el anidamiento. No es posible manejar una profundidad arbitraria de anidamiento. Perl 5.6 contiene una característica experimental que permite que las expresiones regulares sean recursivas (entre otras cosas). El elemento especial (?R) está disponible para el caso específico de recursión. Este patrón de PCRE resuelve el problema de los paréntesis (asumiendo que la opción PCRE_EXTENDED es definida, de modo que los espacios en blanco sean ignorados): *\(((?>[^()]+) | (?R))* \)*

Primero coincide con un paréntesis de apertura. Luego coincide con cualquier número de subcadenas, que pueden ser o bien una secuencia de no-paréntesis, o una coincidencia recursiva del patrón mismo (es decir, una subcadena correctamente envuelta por paréntesis). Finalmente hay un

paréntesis de cierre.

Este patrón de ejemplo en particular contiene repeticiones anidadas ilimitadas, así que el uso de un sub-patrón de una aplicación para la comparación de cadenas de no-paréntesis es importante cuando se aplica el patrón a cadenas que no coinciden. Por ejemplo, cuando se aplica a *(aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa()* descubre una "no coincidencia" rápidamente. Sin embargo, si un sub-patrón de una aplicación no es usado, la comparación es ejecutada por un tiempo muy largo realmente ya que hay muchas formas diferentes en las que las repeticiones + y * pueden moldear el asunto, y todas deben ser probadas antes que pueda reportarse el fallo.

Los valores establecidos para cualquier sub-patrón de captura son aquellos del nivel de recursión más externo en el que esté definido el valor del sub-patrón. Si el patrón anterior fuera comparado contra *(ab(cd)ef)* el valor para los paréntesis de captura es "ef", el cual es el último valor tomado del nivel superior. Si se agregan paréntesis adicionales, produciendo *\((((?>[^()]+) | (?R))*) \)* entonces la cadena que capturan es "ab(cd)ef", los contenidos de los paréntesis del nivel superior. Si hay más de 15 paréntesis de captura en un patrón, PCRE tiene que obtener memoria extra para almacenar datos durante una recursión, cosa que logra usando pcre_malloc, y la libera más adelante mediante pcre_free. Si no se puede obtener memoria, guarda datos únicamente para los primeros 15 paréntesis de captura, dado que no hay forma de producir un error de memoria-insuficiente desde el interior de una recursión.

Rendimientos

Ciertos elementos que pueden aparecer entre los patrones son más eficientes que otros. Es más eficiente usar una clase de caracteres como [aeiou] que un conjunto de alternativas tal como (a|e|i|o|u). En general, la contrucción más simple que ofrezca el comportamiento requerido es usualmente la más eficiente. El libro de Jeffrey Friedl contiene una cantidad considerable de comentarios sobre la optimización de expresiones regulares para un rendimiento eficiente.

Cuando un patrón empieza con .* y la opción PCRE_DOTALL está definida, el patrón es anclado implícitamente por PCRE, ya que sólo puede coincidir al inicio de la cadena de asunto. Sin embargo, si PCRE_DOTALL no es definido, PCRE no puede hacer esta optimización, ya que el meta-caracter . no coincide entonces con una nueva línea y si la cadena de asunto contiene nuevas líneas, el patrón podría coincidir desde el caracter inmediatamente siguiente a una de ellas en vez del inicio absoluto. Por ejemplo, el patrón *(.*) second* coincide con el asunto "first\nand second" (en donde \n representa un caracter de nueva línea) con la primera subcadena capturada que sea "and". Para conseguirlo, PCRE tiene que reintentar la coincidencia comenzando en cada nueva línea del asunto.

Si está usando un patrón así con cadenas de asunto que no contienen nuevas líneas, el mejor rendimiento se obtiene definiendo PCRE_DOTALL o iniciando el patrón con ^.* para indicar un anclamiento explícito. Esto le ahorra a PCRE tener que examinar toda la cadena de entrada buscando nuevas líneas para empezar de nuevo.

Tenga cuidado con los patrones que contienen repeticiones anidadas ilimitadas. Éstos pueden tomar un tiempo de ejecución muy largo cuando se aplican sobre una cadena que no coincide. Considere el fragmento de patrón *(a+)**

Éste puede coincidir con "aaaa" en 33 maneras diferentes, y este número crece muy rápidamente a medida que la cadena se hace más larga. (La repetición * puede coincidir 0, 1, 2, 3, o 4 veces, y por cada uno de esos casos diferentes a 0, las repeticiones + pueden coincidir un número diferente de veces.) Cuando el resto del patrón es de tal forma que la coincidencia completa está destinada a fallar, PCRE en principio intenta cada variación posible, y esto puede tomar un tiempo

extremadamente largo.

Una optimización atrapa algunos de los casos más simples tales como *(a+)*b* en donde un caracter literal se encuentra a continuación. Antes de embarcarse en el procedimiento estándar de comparación, PCRE chequea que haya una letra "b" más adelante en la cadena de asunto, y si no lo hay, falla la coincidencia inmediatamente. Sin embargo, cuando no hay un literal a continuación, esta optimización no puede ser usada. Puede apreciar la diferencia al comparar el comportamiento de *(a+)*\d* con el patrón anterior. El primero produce un fallo casi instantáneamente cuando se aplica a una línea completa de caracteres "a", mientras que el segundo toma un tiempo apreciable con cadenas más largas de aproximadamente 20 caracteres.

preg_grep

(PHP 4 , PHP 5)

preg_grep -- Devuelve un array con los elementos que casen con el patrón

Descripción

array **preg_grep** (string pattern, array input)

preg_grep() devuelve un array conteniendo los elementos del array *input* que emparejen con el patrón (*pattern*) dado.

Ejemplo 1. Ejemplo de la función preg_grep()

```
preg_grep("/^(\d+)?\.\d+$/", $array); // encuentra todos los números reales en el array
```

Nota: Esta función fue añadida en PHP 4.0.

preg_match_all

(PHP 3>= 3.0.9, PHP 4 , PHP 5)

preg_match_all -- Realizar una comparación global con una expresión regular

Descripción

int **preg_match_all** (string patron, string asunto, array &coincidencias [, int banderas [, int desplazamiento]])

Busca el *asunto* por todas las coincidencias con la expresión regular dada en *patron*, y las coloca en *coincidencias* en el orden especificado por *banderas*.

Después de que la primera coincidencia es encontrada, las búsquedas subsiguientes continúan desde el final de la última coincidencia.

banderas puede ser una combinación de las siguientes banderas (note que no tiene sentido usar **PREG_PATTERN_ORDER** junto con **PREG_SET_ORDER**):

PREG_PATTERN_ORDER

Ordena los resultados de tal forma que $coincidencias[0] es una matriz con las coincidencias

completas del patrón, $coincidencias[1] es una matriz con las cadenas que coinciden con el primer sub-patrón entre paréntesis, y así sucesivamente.

```
<?php
preg_match_all("|<[^>]+>(.*)</[^>]+>|U",
    "<b>ejemplo: </b><div align=left>esta es una prueba</div>",
    $salida, PREG_PATTERN_ORDER);
echo $salida[0][0] . ", " . $salida[0][1] . "\n";
echo $salida[1][0] . ", " . $salida[1][1] . "\n";
?>
```

Este ejemplo producirá:

```
<b>ejemplo: </b>, <div align=left>esta es una prueba</div>
ejemplo: , esta es una prueba
```

Así que $salida[0] contiene una matriz de cadenas que coincidieron con el patrón completo, y $salida[1] contiene una matriz de cadenas ubicadas entre etiquetas.

PREG_SET_ORDER

Ordena los resultados de forma tal que $coincidencias[0] es una matriz que contiene el primer conjunto de coincidencias, $coincidencias[1] es una matriz con el segundo conjunto de coincidencias, y así sucesivamente.

```
<?php
preg_match_all("|<[^>]+>(.*)</[^>]+>|U",
    "<b>ejemplo: </b><div align=\"left\">esta es una prueba</div>",
    $salida, PREG_SET_ORDER);
echo $salida[0][0] . ", " . $salida[0][1] . "\n";
echo $salida[1][0] . ", " . $salida[1][1] . "\n";
?>
```

Este ejemplo producirá:

```
<b>ejemplo: </b>, ejemplo:
<div align="left">esta es una prueba</div>, esta es una prueba
```

En este caso, $coincidencias[0] es el primer conjunto de coincidencias, y $coincidencias[0][0] tiene el texto que coincidió con el patrón completo, $coincidencias[0][1] tiene el texto que coincidió con el primer sub-patrón y así sucesivamente. De forma semejante, $coincidencias [1] es el segundo conjunto de coincidencias, etc.

PREG_OFFSET_CAPTURE

Si es pasada esta bandera, para cada coincidencia que ocurre, será devuelto también el desplazamiento de la cadena adjunta. Note que esto modifica el valor de retorno, convirtiéndolo en una matriz en donde cada elemento es una matriz que consiste de la cadena que coincidió en la posición *0*, y su desplazamiento de cadena al interior del *asunto* en la posición *1*. Esta bandera se encuentra disponible a partir de PHP 4.3.0.

Si no se indica bandera alguna, se asume el uso de **PREG_PATTERN_ORDER**.

Normalmente, la búsqueda comienza desde el inicio de la cadena de asunto. El parámetro opcional *desplazamiento* puede ser usado para especificar el lugar alternativo desde donde debe iniciar la búsqueda. Es ñalente a pasar **substr()**($asunto, $desplazamiento) a **preg_match()** en lugar de la cadena de asunto. El parámetro *desplazamiento* se encuentra disponible a partir de PHP 4.3.3.

Devuelve el número de coincidencias con el patrón completo (que puede ser cero), o **FALSE** si ocurre un error.

Ejemplo 1. Obtener todos los números telefónicos de un segmento de texto.

```
<?php
preg_match_all("/\(?  (\d{3})?  \)?  (?(1)  [\-\s] ) \d{3}-\d{4}/x",
               "Llame al 555-1212  1-800-555-1212", $telefonos);
?>
```

Ejemplo 2. Encontrar etiquetas HTML coincidentes (de forma ambiciosa)

```
<?php

// El \\2 es un ejemplo de referencia hacia atras. Este le dice a pcre
// que debe buscar el segundo conjunto de parentesis en la expresion
// regular misma, que seria ([\w]+) en este caso. La barra invertida
// extra es requerida ya que la cadena se encuentra entre comillas
// dobles.
$html = "<b>texto en negrilla</b><a href=hola.html>haga clic aqui</a>";

preg_match_all("/(<([\w]+)[^>]*>)(.*)(<\/\\2>)/", $html, $coincidencias);

for ($i=0; $i< count($coincidencias[0]); $i++) {
  echo "coincidencia: " . $coincidencias[0][$i] . "\n";
  echo "parte 1: " . $coincidencias[1][$i] . "\n";
  echo "parte 2: " . $coincidencias[3][$i] . "\n";
  echo "parte 3: " . $coincidencias[4][$i] . "\n\n";
}
?>
```

Este ejemplo producirá

```
coincidencia: <b>texto en negrilla</b>
parte 1: <b>
parte 2: texto en negrilla
parte 3: </b>

coincidencia: <a href=hola.html>haga clic aqui</a>
parte 1: <a href=hola.html>
parte 2: haga clic aqui
parte 3: </a>
```

Vea también **preg_match()**, **preg_replace()**, y **preg_split()**.

preg_match

(PHP 3>= 3.0.9, PHP 4 , PHP 5)

preg_match -- Realiza un emparejamiento dada una expresión

Descripción

int **preg_match** (string pattern, string subject [, array matches])

Busca en *subject* para un emparejamiento, dada la expresión *pattern*.

Si *matches* es dado, entonces será definido con el resultado de la búsqueda. $matches[0] contendrá el texto que empareja con el patrón en su totalidad. $matches[1] tendrá la cadena que empareje con el primer subpatrón que esté entre paréntesis y así sucesivamente.

Devuelve **`TRUE`** si se encontró en la cadena un emparejamiento dado el patrón *pattern*, **`FALSE`** si no se produjo o hubo un error.

Ejemplo 1. Obtener el número de la siguiente página dada una cadena

```
if (preg_match("/page\s+#(\d+)/i", "Go to page #9.", $parts))
    print "Next page is $parts[1]";             // La siguiente página es $parts[1]
else
    print "Page not found.";                    // Página no encontrada
```

Examinar también **preg_match_all()**, **preg_replace()**, y **preg_split()**.

preg_quote

(PHP 3>= 3.0.9, PHP 4 , PHP 5)

preg_quote -- Prepara los caracteres de expresiones

Descripción

string **preg_quote** (string str)

preg_quote() toma *str* y pone una barra invertida (\) delante de todo carácter que sea parte de la sintaxis de las expresiones. Es útil si tienes una cadena en tiempo de ejecución y puede contener caracteres especiales.

Los caracteres especiales de las expresiones son:

```
. \\ + * ? [ ^ ] $ ( ) { } = ! < > | :
```

Nota: Esta función fue añadida en PHP 3.0.9.

preg_replace_callback

(PHP 4 >= 4.0.5, PHP 5)

preg_replace_callback -- Realizar una búsqueda con expresiones regulares y generar reemplazos usando una llamada de retorno

Descripción

mixed **preg_replace_callback** (mixed patron, callback llamada_de_retorno, mixed fuente [, int limite])

El comportamiento de esta función es casi idéntico al de **preg_replace()**, con la excepción de que en lugar del parámetro *reemplazo*, uno debe especificar una *llamada_de_retorno* que será usada pasándole una matriz de los elementos coincidentes en la cadena fuente. La llamada de retorno debería devolver la cadena de reemplazo.

Ejemplo 1. Ejemplo de preg_replace_callback()

```
<?php
  // este texto fue usado en 2002
  // queremos actualizarlo para 2003

  $texto = "El d&iacute;a de los inocentes es 04/01/2002\n";
  $texto.= "La &uacute;ltima navidad fue 12/24/2001\n";


  // la llamada de retorno

  function siguiente_anyo($coincidencias)
  {
    // como es usual: $coincidencias[0] es la coincidencia completa
    // $coincidencias[1] la coincidencia para el primer subpatron
    // ubicado entre '(...)' y asi sucesivamente

    return $coincidencias[1].($coincidencias[2]+1);
  }

  echo preg_replace_callback(
              "|(\d{2}/\d{2}/)(\d{4})|",
              "siguiente_anyo",
              $texto);

  // el resultado es:
  // El d&iacute;a de los inocentes es 04/01/2003
  // La &uacute;ltima navidad fue 12/24/2002
?>
```

Con frecuencia necesitará la función de *llamada de retorno* cuando use **preg_replace_callback()** únicamente en un lugar. En este caso, usted puede usar **create_function()** para declarar una función anónima como llamada de retorno dentro del llamado a **preg_replace_callback()**. Al hacerlo de este modo, usted tendrá toda la información necesaria para el llamado en un solo lugar y no abarrota su espacio de nombres de funciones con nombres de llamadas de retorno que no son usadas en ninguna otra parte.

Ejemplo 2. preg_replace_callback() y create_function()

```
<?php
  /* un filtro de linea de comandos tipo Unix para convertir letras
   * mayusculas al comienzo de los paragrafos a minusculas */

$da = fopen("php://stdin", "r") or die("no se puede leer stdin");

while (!feof($da)) {
    $linea = fgets($da);
    $linea = preg_replace_callback(
        '|<p>\s*\w|',
        create_function(
            // las comillas sencillas son cruciales aqui,
            // o alternativamente escapar todos los signos $ como \$
            '$coincidencias',
            'return strtolower($coincidencias[0]);'
            ),
        $linea
        );
    echo $linea;
}
fclose($da);

?>
```

Vea también **preg_replace()** y **create_function()**.

preg_replace

(PHP 3>= 3.0.9, PHP 4 , PHP 5)

preg_replace -- Lleva a cabo la búsqueda de una expresión y su sustitución

Descripción

mixed **preg_replace** (mixed patron, mixed reemplazo, mixed entrada [, int limite])

Busca en *entrada* los emparejamientos con *patron* y los sustituye por *reemplazo*. Si se especifica el parámetro *limite*, solo *limite* serán reemplazados; si *limite* se omite o es igual a -1 todas las coincidencias serán reemplazadas.

reemplazo puede contener referencias de la forma *n*. Éstas serán sustituidas por el texto obtenido por el patrón del paréntesis *n*ésimo. *n* puede tener un valor de cero a noventa y nueve, y *0* se refiere al texto casado por el patrón completo. Para obtener el número del subpatrón de búsqueda, los paréntesis abiertos son contados de izquierda derecha tomando el primero como uno.

Cuando trabajes con patrones de reemplazo donde una referencia hacia atrás esté inmediatamente seguida por otro número (por ejemplo: poniendo un número literal después de un patrón coincidente), no puedes usar el método normal \\1 para la referencia hacia atrás. \\11, por ejemplo, confundirá a **preg_replace()()** ya que no sabe si quieres usar la referencia hacia atrás \\1 seguida de un 1 literal, o la referencia \\11 seguida de nada. En este caso la solución es usar \${1}1. Esto usa la referencia \\1, dejando el 1 literal tras el.

Ejemplo 1. Usando referencias hacia atrás seguidas de números literales

```
<?php
$cadena = "Abril 15, 2003";
$patron = "/(\w+) (\d+), (\d+)/i";
$reemplazo = "\${1}1,\$3";
echo preg_replace($patron, $reemplazo, $cadena);
?>
```

Salida:

```
Abril1,2003
```

Si el patrón no es encontrado en *entrada*, entonces no se realizarán cambios.

Todos los parámetros de la función **preg_replace()** (excepto *limite*) pueden ser un array unidimensional. Cuando se usan arrays como *patron* o *reemplazo*, las claves son procesadas en el orden en que aparecen en el array. Este *no es necesariamente* el orden numérico del mismo. Si usas indices numéricos para identificar que *patron* corresponde a cada *reemplazo* deberás aplicar un **ksort()** a cada uno de los arrays antes de aplicar **preg_replace()**.

Ejemplo 2. Usando arrays con claves numéricas con preg_replace()

```
<?php
$cadena = "El veloz murciélago hindú comía feliz cardillo y kiwi.";

$patrones[0] = "/veloz/";
$patrones[1] = "/hind&uacute;/";
$patrones[2] = "/murci&eacute;lago/";

$reemplazos[2] = "oso";
$reemplazos[1] = "negro";
$reemplazos[0] = "lento";

echo preg_replace($patrones, $reemplazos, $cadena);
?>
```

Salida:

```
El oso lento negro com&iacute;a feliz cardillo y kiwi.
```

Aplicando **ksort()** sobre los patrones y los reemplazos podemos obtener el comportamiento esperado.

```
<?php

ksort($patrones);
ksort($reemplazos);

echo preg_replace($patrones, $reemplazos, $cadena);

?>
```

Salida:

```
El lento oso negro comía feliz cardillo y kiwi.
```

Si *entrada* es un array, entonces la búsqueda y sustitución es realizada para todos los elementos de *entrada*, y el valor devuelto es también un array.

Si *patron* y *reemplazo* son arrays, entonces **preg_replace()** toma un valor desde cada array y los usas para buscar y sustituir sobre *entrada*. Si *reemplazo* tiene menos valores que *patron*, entonces la cadena vacía es usada como valor para el resto de sustituciones. Si *patron* es una array y *reemplazo* es una cadena, entonces esta cadena de sustitución es usada para todos los valores de *patron*. Sin embargo, lo contrario no tiene sentido.

El modificador */e* hace que la función **preg_replace()** trate el parámetro *reemplazo* como código PHP después de que la apropiada sustitución sea hecha. Atención, asegúrate que *reemplazo* es un código PHP correcto, de otro modo PHP dará un error de parse en la línea que contenga **preg_replace()**.

> **Nota:** Este modificador fue añadido en PHP 4.0.

Ejemplo 3. Sustituir varios valores

```
<?php
$patrones = array("/(19|20\d{2})-(\d{1,2})-(\d{1,2})/", "/^\s*{(\w+)}\s*=/");
$replace = array("\\3/\\4/\\1", "$\\1 =");
print preg_replace($patrones, $replace, "{startDate} = 1999-5-27");
?>
```

Esta ejemplo dará como resultado:

```
$startDate = 5/27/1999
```

Ejemplo 4. Usar el modificador /e

```
<?php
preg_replace("/(<\/?)(\w+)([^>]*>)/e", "'\\1'.strtoupper('\\2').'\\3'", $html_body);
?>
```

Pondrá en mayúscula todos los tags HTML del texto de entrada.
Examina también **preg_match()**, **preg_match_all()**, y **preg_split()**.

preg_split

(PHP 3>= 3.0.9, PHP 4 , PHP 5)

preg_split -- Divide una cadena dada una expresión

Descripción

array **preg_split** (string pattern, string subject [, int limit [, int flags]])

> **Nota:** El parámetro *flags* fue añadido en la Beta 3 de PHP

Devuelve un array conteniendo las subcadenas de *subject* divididas mediante los emparejamientos limitados por *pattern*.

Si *limit* es proporcionado, entonces sólo *limit* subcadenas son devueltas.

Si el flags es PREG_SPLIT_NO_EMPTY entonces las cadenas vacías no serán devueltas por **preg_split()**.

Ejemplo 1. Obtener las partes de una cadena de búsqueda

```
$keywords = preg_split("/[\s,]+/", "hypertext language, programming");
```

Examinar también **preg_match()**, **preg_match_all()**, y **preg_replace()**.

XCVII. Funciones PDF

Introducción

Las funciones PDF en PHP pueden crear archivos PDF utilizando la biblioteca PDFlib creada por Thomas Merz.

La documentación en esta sección solamente es una descripción de las funciones de la biblioteca PDFlib y no debería considerarse una referencia exhaustiva. Se ha de consultar la documentación incluida en el código fuente de la distribución de PDFlib para una completa y detallada explicación de cada función. Proporciona muy buena descripción de las capacidades de PDFlib y contiene actualizada la documentación de todas las funciones.

Todas las funciones de PDFlib y del módulo PHP tienen nombres iguales para las funciones y parámetros. Se necesitará entender algunos de los conceptos básicos de PDF y PostScript para un eficiente uso de esta extensión. Todas las longitudes y coordenadas se mesuran en puntos PostScript. Generalmente hay 72 puntos PostScript por pulgada, pero esto depende de la resolución de salida. Se puede consultar la documentación incluida en la distribución de PDFlib para una detallada explicación del sistema de coordenadas utilizado.

Hay que tener en cuenta que la mayoría de las funciones PDF requieren un primer parámetro *pdfdoc*. En los siguientes ejemplos hay más información.

> **Nota:** Si se está interesado en alternativas de generadores gratis de PDF que no utilizen liberías externas PDF, mirar este FAQ relacionado.

Requirimientos

PDFlib está disponible para descargar en http://www.pdflib.com/products/pdflib/index.html, pero requiere la compra de una licencia para uso comercial. Se requieren las bibliotecas JPEG y TIFF para compilar esta extensión.

Compatibilidad con versiones antiguas de PDFlib

Cualquier versión de PHP después del 9 de Marzo del 2000 no soporta versiones de PDFlib anteriores a la 3.0.

PDFlib 3.0 o superior es compatible desde PHP 3.0.19 en adelante.

Instalación

Esta extension PECL no esta ligada a PHP. Mas informacion sobre nuevos lanzamientos, descargas ficheros de fuentes, informacion sobre los responsables asi como un 'CHANGELOG', se puede encontrar aqui: http://pecl.php.net/package/pdflib.

To get these functions to work in PHP < 4.3.9, you have to compile PHP with *--with-pdflib[=DIR]*. DIR is the PDFlib base install directory, defaults to `/usr/local`. In addition you can specify the jpeg, tiff, and pnglibrary for PDFlib to use, which is optional for PDFlib 4.x. To do so add to your configure line the options *--with-jpeg-dir[=DIR] --with-png-dir[=DIR] --with-tiff-dir[=DIR]*.

When using version 3.x of PDFlib, you should configure PDFlib with the option *--enable-shared-pdflib*.

As of PHP 4.3.9, you must install this extension through PEAR, using the following command: **pear install pdflib**.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Confusiones con antiguas versiones de PDFlib

Desde PHP 4.0.5, la extensión PHP para PDFlib es oficialmente soportada por PDFlib GmbH. Esto significa que todas las funciones descritas en el manual de PDFlib (V3.00 o superior) son soportadas por PHP 4 con el mismo funcionamiento y parámetros. Sólo los valores devueltos pueden variar en el manual PDFlib, ya que PHP adoptó la convención de devolver **`FALSE`**. Por razones de compatibilidad, PDFlib aún soporta las antiguas funciones, pero deberían reemplazarlas en sus nuevas versiones. PDFlib GmbH no dará soporte a cualquier problema causado por el uso de estas funciones obsoletas.

Tabla 1. Funciones obsoletas y sus reemplazos.

Antigua función	Reemplazo
pdf_put_image()	Ya no se necesita.
pdf_execute_image()	Ya no se necesita.
pdf_get_annotation()	**pdf_get_bookmark()** utilizando los mismos parámetros.
pdf_get_font()	**pdf_get_value()** pasando *"font"* como segundo parámetro.
pdf_get_fontsize()	**pdf_get_value()** pasando *"fontsize"* como segundo parámetro.
pdf_get_fontname()	**pdf_get_parameter()** pasando *"fontname"* como segundo parámetro.
pdf_set_info_creator()	**pdf_set_info()** pasando *"Creator"* como segundo parámetro.

Antigua función	Reemplazo
pdf_set_info_title()	pdf_set_info() pasando *"Title"* como segundo parámetro.
pdf_set_info_subject()	pdf_set_info() pasando *"Subject"* como segundo parámetro.
pdf_set_info_author()	pdf_set_info() pasando *"Author"* como segundo parámetro.
pdf_set_info_keywords()	pdf_set_info() pasando *"Keywords"* como segundo parámetro.
pdf_set_leading()	pdf_set_value() pasando *"leading"* como segundo parámetro.
pdf_set_text_rendering()	pdf_set_value() pasando *"textrendering"* como segundo parámetro.
pdf_set_text_rise()	pdf_set_value() pasando *"textrise"* como segundo parámetro.
pdf_set_horiz_scaling()	pdf_set_value() pasando *"horizscaling"* como segundo parámetro.
pdf_set_text_matrix()	Ya no se necesita.
pdf_set_char_spacing()	pdf_set_value() pasando *"charspacing"* como segundo parámetro.
pdf_set_word_spacing()	pdf_set_value() pasando *"wordspacing"* como segundo parámetro.
pdf_set_transition()	pdf_set_parameter() pasando *"transition"* como segundo parámetro.
pdf_open()	pdf_new() más la subsecuente llamada de pdf_open_file()
pdf_set_font()	pdf_findfont() más la subsecuente llamada de pdf_setfont()
pdf_set_duration()	pdf_set_value() pasando *"duration"* como segundo parámetro.
pdf_open_gif()	pdf_open_image_file() pasando *"gif"* como segundo parámetro.
pdf_open_jpeg()	pdf_open_image_file() pasando *"jpeg"* como segundo parámetro.
pdf_open_tiff()	pdf_open_image_file() pasando *"tiff"* como segundo parámetro.
pdf_open_png()	pdf_open_image_file() pasando *"png"* como segundo parámetro.
pdf_get_image_width()	pdf_get_value() pasando *"imagewidth"* como segundo parámetro y la imágen como tercer parámetro.
pdf_get_image_height()	pdf_get_value() pasando *"imageheight"* como segundo parámetro y la imágen como tercer parámetro.

Ejemplos

La mayoría de las funciones son bastante fáciles de utilizar. La parte más difícil probablemente es la creación de un primer documento PDF. El siguiente ejemplo debería ayudar para comenzar. El ejemplo crea el archivo `test.pdf` en una página. La página contiene el texto "Times Roman outlined" en un contorno, con fuente de 30pt. El texto también está subrayado.

Ejemplo 1. Creando un documento PDF con PDFlib

```
<?php
$pdf = pdf_new();
pdf_open_file($pdf, "test.pdf");
pdf_set_info($pdf, "Author", "Javier Tacon");
pdf_set_info($pdf, "Title", "Test for PHP wrapper of PDFlib 2.0");
pdf_set_info($pdf, "Creator", "See Author");
pdf_set_info($pdf, "Subject", "Testing");
pdf_begin_page($pdf, 595, 842);
pdf_add_outline($pdf, "Page 1");
$font = pdf_findfont($pdf, "Times New Roman", "winansi", 1);
pdf_setfont($pdf, $font, 10);
pdf_set_value($pdf, "textrendering", 1);
pdf_show_xy($pdf, "Times Roman outlined", 50, 750);
pdf_moveto($pdf, 50, 740);
pdf_lineto($pdf, 330, 740);
pdf_stroke($pdf);
pdf_end_page($pdf);
pdf_close($pdf);
pdf_delete($pdf);
echo "<A HREF=getpdf.php>finished</A>";
?>
```

El script `getpdf.php` simplemente devuelve el documento pdf.

Ejemplo 2. Mostrando un documento PDF precalculado

```
<?php
$len = filesize($filename);
header("Content-type: application/pdf");
header("Content-Length: $len");
header("Content-Disposition: inline; filename=foo.pdf");
readfile($filename);
?>
```

La distrubución PDFlib contiene un ejemplo más complejo para crear un reloj analógico en una página. Aquí se utiliza el método de creación en memoria de PDFlib para no tener que crear un archivo temporal. El ejemplo se ha convertido a PHP desde uno de PDFlib (El mismo ejemplo está disponible en la documentación ClibPDF.)

Ejemplo 3. Ejemplo pdfclock de la distribución PDFlib

```
<?php
$radius = 200;
$margin = 20;
$pagecount = 10;

$pdf = pdf_new();

if (!pdf_open_file($pdf, "")) {
    echo error;
    exit;
};

pdf_set_parameter($pdf, "warning", "true");

pdf_set_info($pdf, "Creator", "pdf_clock.php");
pdf_set_info($pdf, "Author", "Uwe Steinmann");
pdf_set_info($pdf, "Title", "Analog Clock");

while ($pagecount-- > 0) {
    pdf_begin_page($pdf, 2 * ($radius + $margin), 2 * ($radius + $margin));

    pdf_set_parameter($pdf, "transition", "wipe");
    pdf_set_value($pdf, "duration", 0.5);

    pdf_translate($pdf, $radius + $margin, $radius + $margin);
    pdf_save($pdf);
    pdf_setrgbcolor($pdf, 0.0, 0.0, 1.0);

    /* minute strokes */
    pdf_setlinewidth($pdf, 2.0);
    for ($alpha = 0; $alpha < 360; $alpha += 6) {
        pdf_rotate($pdf, 6.0);
        pdf_moveto($pdf, $radius, 0.0);
        pdf_lineto($pdf, $radius-$margin/3, 0.0);
        pdf_stroke($pdf);
    }

    pdf_restore($pdf);
    pdf_save($pdf);

    /* 5 minute strokes */
    pdf_setlinewidth($pdf, 3.0);
    for ($alpha = 0; $alpha < 360; $alpha += 30) {
        pdf_rotate($pdf, 30.0);
        pdf_moveto($pdf, $radius, 0.0);
        pdf_lineto($pdf, $radius-$margin, 0.0);
        pdf_stroke($pdf);
    }

    $ltime = getdate();

    /* draw hour hand */
    pdf_save($pdf);
    pdf_rotate($pdf,-(($ltime['minutes']/60.0)+$ltime['hours']-3.0)*30.0);
    pdf_moveto($pdf, -$radius/10, -$radius/20);
    pdf_lineto($pdf, $radius/2, 0.0);
    pdf_lineto($pdf, -$radius/10, $radius/20);
    pdf_closepath($pdf);
    pdf_fill($pdf);
    pdf_restore($pdf);

    /* draw minute hand */
    pdf_save($pdf);
    pdf_rotate($pdf,-(($ltime['seconds']/60.0)+$ltime['minutes']-15.0)*6.0);
    pdf_moveto($pdf, -$radius/10, -$radius/20);
    pdf_lineto($pdf, $radius * 0.8, 0.0);
    pdf_lineto($pdf, -$radius/10, $radius/20);
    pdf_closepath($pdf);
    pdf_fill($pdf);
    pdf_restore($pdf);

    /* draw second hand */
```

Ver también

Nota: Una alternativa de módulo PHP para la creación de documentos PDF basados en FastIO's ClibPDF está disponible. Mirar la sección ClibPDF para más detalles. Tener en cuenta que ClibPDF tiene alguna diferencia con PDFlib.

Tabla de contenidos

pdf_get_pdi_value -- Gets some PDI numerical parameters
PDF_get_value -- Gets certain numerical value
pdf_initgraphics -- Resets graphic state
PDF_lineto -- Draws a line
pdf_makespotcolor -- Makes a spotcolor
PDF_moveto -- Sets current point
pdf_new -- Creates a new pdf resource
pdf_open_ccitt -- Opens a new image file with raw CCITT data
pdf_open_file -- Opens a new pdf object
PDF_open_gif -- Opens a GIF image
pdf_open_image_file -- Reads an image from a file
pdf_open_image -- Versatile function for images
PDF_open_jpeg -- Opens a JPEG image
PDF_open_memory_image -- Opens an image created with PHP's image functions
pdf_open_pdi_page -- Prepare a page
pdf_open_pdi -- Opens a PDF file
PDF_open_png -- Opens a PNG image
pdf_open_tiff -- Deprecated: Opens a TIFF image
PDF_open -- Opens a new pdf document
PDF_place_image -- Places an image on the page
pdf_place_pdi_page -- Places an image on the page
PDF_rect -- Draws a rectangle
PDF_restore -- Restores formerly saved environment
PDF_rotate -- Sets rotation
PDF_save -- Saves the current environment
PDF_scale -- Sets scaling
PDF_set_border_color -- Sets color of border around links and annotations
PDF_set_border_dash -- Sets dash style of border around links and annotations
PDF_set_border_style -- Sets style of border around links and annotations
PDF_set_char_spacing -- Sets character spacing
PDF_set_duration -- Sets duration between pages
PDF_set_font -- Selects a font face and size
PDF_set_horiz_scaling -- Sets horizontal scaling of text
pdf_set_info_author -- Deprecated: Fills the author field of the document
pdf_set_info_creator -- Deprecated: Fills the creator field of the document
pdf_set_info_keywords -- Deprecated: Fills the keywords field of the document
pdf_set_info_subject -- Deprecated: Fills the subject field of the document
pdf_set_info_title -- Deprecated: Fills the title field of the document
PDF_set_info -- Fills a field of the document information
PDF_set_leading -- Sets distance between text lines
PDF_set_parameter -- Sets certain parameters
PDF_set_text_matrix -- Sets the text matrix
PDF_set_text_pos -- Sets text position
PDF_set_text_rendering -- Determines how text is rendered
PDF_set_text_rise -- Sets the text rise
PDF_set_value -- Sets certain numerical value
PDF_set_word_spacing -- Sets spacing between words
pdf_setcolor -- Sets fill and stroke color
PDF_setdash -- Sets dash pattern
PDF_setflat -- Sets flatness
pdf_setfont -- Set the current font
PDF_setgray_fill -- Sets filling color to gray value
PDF_setgray_stroke -- Sets drawing color to gray value
PDF_setgray -- Sets drawing and filling color to gray value
PDF_setlinecap -- Sets linecap parameter

PDF_setlinejoin -- Sets linejoin parameter
PDF_setlinewidth -- Sets line width
pdf_setmatrix -- Sets current transformation matrix
PDF_setmiterlimit -- Sets miter limit
pdf_setpolydash -- Deprecated: Sets complicated dash pattern
PDF_setrgbcolor_fill -- Sets filling color to rgb color value
PDF_setrgbcolor_stroke -- Sets drawing color to rgb color value
PDF_setrgbcolor -- Sets drawing and filling color to rgb color value
PDF_show_boxed -- Output text in a box
PDF_show_xy -- Output text at given position
PDF_show -- Output text at current position
PDF_skew -- Skews the coordinate system
PDF_stringwidth -- Returns width of text using current font
PDF_stroke -- Draws line along path
PDF_translate -- Sets origin of coordinate system

pdf_add_annotation

(PHP 3>= 3.0.12, PHP 4 , PHP 5)

pdf_add_annotation -- Adds annotation

Description

void **pdf_add_annotation** (int pdf document, double llx, double lly, double urx, double ury, string title, string content)

The **pdf_add_annotation()** adds a note with the lower left corner at (*llx*, *lly*) and the upper right corner at (*urx*, *ury*).

pdf_add_bookmark

(PHP 4 >= 4.0.1, PHP 5)

pdf_add_bookmark -- Adds bookmark for current page

Description

int **pdf_add_bookmark** (resource pdfdoc, string text, int parent, int open)

Add a nested bookmark under *parent*, or a new top-level bookmark if *parent* = 0. Returns a bookmark descriptor which may be used as parent for subsequent nested bookmarks. If *open* = 1, child bookmarks will be folded out, and invisible if *open* = 0. Parameters *parent* and *open* were optional before PHP 5.

Ejemplo 1. pdf_add_bookmark() example

```php
<?php
// create a new PDF

$pdf = pdf_new();
pdf_open_file($pdf);
pdf_set_info($pdf, "Author", "Bob Nijman");

// begin a new page
pdf_begin_page($pdf, 300, 300);

// add a top-level bookmark
$bookmark = pdf_add_bookmark($pdf, "People");

// add a nested bookmark
pdf_add_bookmark($pdf, "Rasmus", $bookmark);

// and some text
pdf_set_font($pdf, "Helvetica", 20, "host");
$text = "This is R's page";
$width = pdf_stringwidth($pdf, $text);
pdf_set_text_pos($pdf, (300-$width)/2, 100);
pdf_show($pdf, $text);

// close the page and the PDF
pdf_end_page($pdf);
pdf_close($pdf);

?>
```

pdf_add_launchlink

(PHP 4 >= 4.0.5, PHP 5)

pdf_add_launchlink -- Add a launch annotation for current page

Description

bool **pdf_add_launchlink** (resource pdfdoc, float llx, float lly, float urx, float ury, string filename)

Adds a link to a web resource specified by *filename*. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

See also **pdf_add_locallink()**.

pdf_add_locallink

(PHP 4 >= 4.0.5, PHP 5)

pdf_add_locallink -- Add a link annotation for current page

Description

bool **pdf_add_locallink** (resource pdfdoc, float lowerleftx, float lowerlefty, float upperrightx, float upperrighty, int page, string dest)

Add a link annotation to a target within the current PDF file. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

dest is the zoom setting on the destination page, it can be one of *retain*, *fitpage*, *fitwidth*, *fitheight* or *fitbbox*.

See also **pdf_add_launchlink()**.

pdf_add_note

(PHP 4 >= 4.0.5, PHP 5)

pdf_add_note -- Sets annotation for current page

Description

bool **pdf_add_note** (resource pdfdoc, float llx, float lly, float urx, float ury, string contents, string title, string icon, int open)

Sets an annotation for the current page. *icon* is one of *comment*, *insert*, *note*, *paragraph*, *newparagraph*, *key*, or *help*. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

PDF_add_outline

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_add_outline -- Adds bookmark for current page

Description

int **pdf_add_outline** (int pdf document, string text [, int parent [, int open]])

The **PDF_add_outline()** function adds a bookmark with text *text* that points to the current page. The bookmark is inserted as a child of *parent* and is by default open if *open* is not 0. The return value is an identifier for the bookmark which can be used as a parent for other bookmarks. Therefore you can build up hierarchies of bookmarks.

Unfortunately pdflib does not make a copy of the string, which forces PHP to allocate the memory. Currently this piece of memory is not been freed by any PDF function but it will be taken care of by the PHP memory manager.

pdf_add_pdflink

(PHP 3>= 3.0.12, PHP 4 , PHP 5)

pdf_add_pdflink -- Adds file link annotation for current page

Description

bool **pdf_add_pdflink** (resource pdfdoc, float bottom_left_x, float bottom_left_y, float up_right_x, float up_right_y, string filename, int page, string dest)

Add a file link annotation (to a PDF target). Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

See also **pdf_add_locallink()** and **pdf_add_weblink()**.

pdf_add_thumbnail

(PHP 4 >= 4.0.5, PHP 5)

pdf_add_thumbnail -- Adds thumbnail for current page

Description

bool **pdf_add_thumbnail** (resource pdfdoc, int image)

Adds an existing image as thumbnail for the current page. Thumbnail images must not be wider or higher than 106 pixels. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

See also **pdf_open_image()**, **pdf_open_image_file()** and **pdf_open_memory_image()**.

pdf_add_weblink

(PHP 3>= 3.0.12, PHP 4 , PHP 5)

pdf_add_weblink -- Adds weblink for current page

Description

bool **pdf_add_weblink** (resource pdfdoc, float lowerleftx, float lowerlefty, float upperrightx, float upperrighty, string url)

Add a weblink annotation to a target *url* on the Web. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

PDF_arc

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_arc -- Draws an arc

Description

void **pdf_arc** (int pdf document, double x-coor, double y-coor, double radius, double start, double end)

The **PDF_arc()** function draws an arc with center at point (*x-coor*, *y-coor*) and radius *radius*, starting at angle *start* and ending at angle *end*.

See also **PDF_circle()**, **PDF_stroke()**.

pdf_arcn

(PHP 4 >= 4.0.5, PHP 5)

pdf_arcn -- Draws an arc (clockwise)

Description

bool **pdf_arcn** (resource pdfdoc, float x, float y, float r, float alpha, float beta)

Add a clockwise circular arc from *alpha* to *beta* degrees with center (*x*; *y*) and radius *r*. Actual drawing of the circle is performed by the next stroke or fill operation.

Ejemplo 1. pdf_arcn() example

```
<?php

// prepare document
$pdf = pdf_new();
pdf_open_file($pdf, "");
pdf_begin_page($pdf, 595, 842);

// an outlined arcn
pdf_arcn($pdf, 200, 700, 100, 0, 90);
pdf_stroke($pdf);

// a filled arcn
pdf_arcn($pdf, 200, 700, 50, 0, 90);
pdf_fill($pdf);

// an outlined and filled arcn
pdf_setcolor($pdf, "fill", "gray", 0.8);
pdf_arcn($pdf, 400, 700, 50, 0, 90);
pdf_fill_stroke($pdf);

// finish document
pdf_end_page($pdf);
pdf_close($pdf);

header("Content-type: application/pdf");
echo pdf_get_buffer($pdf);

pdf_delete($pdf);
?>
```

See also: **pdf_arc()**, **pdf_circle()**, **pdf_stroke()**, **pdf_fill()** and **pdf_fillstroke()**.

pdf_attach_file

(PHP 4 >= 4.0.5, PHP 5)

pdf_attach_file -- Adds a file attachment for current page

Description

bool **pdf_attach_file** (resource pdfdoc, float llx, float lly, float urx, float ury, string filename, string description, string author, string mimetype, string icon)

Add a file attachment annotation. *icon* is one of *graph*, *paperclip*, *pushpin*, or *tag*. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

> **Nota:** Only the 'Full' Acrobat software will be able to display these file attachments. All other PDF viewers will either show nothing or display a question mark.

PDF_begin_page

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_begin_page -- Starts new page

Description

void **pdf_begin_page** (int pdf document, double width, double height)

The **PDF_begin_page()** function starts a new page with height *height* and width *width*. In order to create a valid document you must call this function and **PDF_end_page()**.

See also **PDF_end_page()**.

pdf_begin_pattern

(PHP 4 >= 4.0.5, PHP 5)

pdf_begin_pattern -- Starts new pattern

Description

int **pdf_begin_pattern** (resource pdfdoc, float width, float height, float xstep, float ystep, int painttype)

Starts a new pattern definition and returns a pattern handle. *width*, and *height* define the bounding box for the pattern. *xstep* and *ystep* give the repeated pattern offsets. *painttype*=1 means that the pattern has its own colour settings whereas a value of 2 indicates that the current colour is used when the pattern is applied.

See also **pdf_end_pattern()**.

pdf_begin_template

(PHP 4 >= 4.0.5, PHP 5)

pdf_begin_template -- Starts new template

Description

int **pdf_begin_template** (resource pdfdoc, float width, float height)

Start a new template definition.

PDF_circle

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_circle -- Draws a circle

Description

void **pdf_circle** (int pdf document, double x-coor, double y-coor, double radius)

The **PDF_circle()** function draws a circle with center at point (*x-coor*, *y-coor*) and radius *radius*.

See also **PDF_arc()**, **PDF_stroke()**.

PDF_clip

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_clip -- Clips to current path

Description

void **pdf_clip** (int pdf document)

The **PDF_clip()** function clips all drawing to the current path.

PDF_close_image

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

PDF_close_image -- Closes an image

Description

void **pdf_close_image** (int image)

The **PDF_close_image()** function closes an image which has been opened with any of the **PDF_open_xxx()** functions.

See also **PDF_open_jpeg()**, **PDF_open_gif()**, **PDF_open_memory_image()**.

pdf_close_pdi_page

(PHP 4 >= 4.0.5, PHP 5)

pdf_close_pdi_page -- Close the page handle

Description

bool **pdf_close_pdi_page** (resource pdfdoc, int pagehandle)

Close the page handle, and free all page-related resources. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

pdf_close_pdi

(PHP 4 >= 4.0.5, PHP 5)

pdf_close_pdi -- Close the input PDF document

Description

bool **pdf_close_pdi** (resource pdfdoc, int dochandle)

Close all open page handles, and close the input PDF document. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

See also **pdf_open_pdi()**.

PDF_close

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_close -- Closes a pdf document

Description

void **pdf_close** (int pdf document)

The **PDF_close()** function closes the pdf document.

> **Nota:** Due to an unclean implementation of the pdflib 0.6 the internal closing of the document also closes the file. This should not be done because pdflib did not open the file, but expects an already open file when **PDF_open()** is called. Consequently it shouldn't close the file. In order to fix this just take out line 190 of the file p_basic.c in the pdflib 0.6 source distribution until the next release of pdflib will fix this.
>
> **Nota:** This function works properly without any patches to pdflib if pdflib 2.0 support is activated.

See also **PDF_open()**, **fclose()**.

PDF_closepath_fill_stroke

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_closepath_fill_stroke -- Closes, fills and strokes current path

Description

void **pdf_closepath_fill_stroke** (int pdf document)

The **PDF_closepath_fill_stroke()** function closes, fills the interior of the current path with the current fill color and draws current path.

See also **PDF_closepath()**, **PDF_stroke()**, **PDF_fill()**, **PDF_setgray_fill()**, **PDF_setgray()**, **PDF_setrgbcolor_fill()**, **PDF_setrgbcolor()**.

PDF_closepath_stroke

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_closepath_stroke -- Closes path and draws line along path

Description

void **pdf_closepath_stroke** (int pdf document)

The **PDF_closepath_stroke()** function is a combination of **PDF_closepath()** and **PDF_stroke()**. It also clears the path.

See also **PDF_closepath()**, **PDF_stroke()**.

PDF_closepath

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_closepath -- Closes path

Description

void **pdf_closepath** (int pdf document)

The **PDF_closepath()** function closes the current path. This means, it draws a line from current point to the point where the first line was started. Many functions like **PDF_moveto()**, **PDF_circle()** and **PDF_rect()** start a new path.

pdf_concat

(PHP 4 >= 4.0.5, PHP 5)

pdf_concat -- Concatenate a matrix to the CTM

Description

bool **pdf_concat** (resource pdfdoc, float a, float b, float c, float d, float e, float f)

Concatenate a matrix to the CTM. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

PDF_continue_text

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_continue_text -- Outputs text in next line

Description

void **pdf_continue_text** (int pdf document, string text)

The **PDF_continue_text()** function outputs the string in *text* in the next line. The distance between the lines can be set with **PDF_set_leading()**.

See also **PDF_show_xy()**, **PDF_set_leading()**, **PDF_set_text_pos()**.

PDF_curveto

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_curveto -- Draws a curve

Description

void **pdf_curveto** (int pdf document, double x1, double y1, double x2, double y2, double x3, double y3)

The **PDF_curveto()** function draws a Bezier curve from the current point to the point (*x3*, *y3*) using (*x1*, *y1*) and (*x2*, *y2*) as control points.

See also **PDF_moveto()**, **PDF_lineto()**, **PDF_stroke()**.

pdf_delete

(PHP 4 >= 4.0.5, PHP 5)

pdf_delete -- Deletes a PDF object

Description

bool **pdf_delete** (resource pdfdoc)

Delete the PDF resource, and free all internal resources. Devuelve **TRUE** si todo se llevó a cabo

correctamente, **FALSE** en caso de fallo.

See also **pdf_new()**.

PDF_end_page

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_end_page -- Ends a page

Description

void **pdf_end_page** (int pdf document)

The **PDF_end_page()** function ends a page. Once a page is ended it cannot be modified anymore.

See also **PDF_begin_page()**.

pdf_end_pattern

(PHP 4 >= 4.0.5, PHP 5)

pdf_end_pattern -- Finish pattern

Description

bool **pdf_end_pattern** (resource pdfdoc)

Finish the pattern definition. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

See also **pdf_begin_pattern()**.

pdf_end_template

(PHP 4 >= 4.0.5, PHP 5)

pdf_end_template -- Finish template

Description

bool **pdf_end_template** (resource pdfdoc)

Finish the template definition. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

PDF_endpath

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_endpath -- Ends current path

Description

void **pdf_endpath** (int pdf document)

The **PDF_endpath()** function ends the current path but does not close it.

See also **PDF_closepath()**.

PDF_fill_stroke

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_fill_stroke -- Fills and strokes current path

Description

void **pdf_fill_stroke** (int pdf document)

The **PDF_fill_stroke()** function fills the interior of the current path with the current fill color and draws current path.

See also **PDF_closepath()**, **PDF_stroke()**, **PDF_fill()**, **PDF_setgray_fill()**, **PDF_setgray()**, **PDF_setrgbcolor_fill()**, **PDF_setrgbcolor()**.

PDF_fill

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_fill -- Fills current path

Description

void **pdf_fill** (int pdf document)

The **PDF_fill()** function fills the interior of the current path with the current fill color.

See also **PDF_closepath()**, **PDF_stroke()**, **PDF_setgray_fill()**, **PDF_setgray()**, **PDF_setrgbcolor_fill()**, **PDF_setrgbcolor()**.

pdf_findfont

(PHP 4 >= 4.0.5, PHP 5)

pdf_findfont -- Prepare font for later use with **pdf_setfont()**

Description

int **pdf_findfont** (resource pdfdoc, string fontname, string encoding, int embed)

Prepare a font for later use with **pdf_setfont()**. The metrics will be loaded, and if *embed* is nonzero, the font file will be checked, but not yet used. *encoding* is one of *builtin*, *macroman*, *winansi*, *host*, a user-defined encoding name or the name of a CMap. Parameter *embed* was optional before PHP 5.

pdf_findfont() returns a font handle or **FALSE** on error.

Ejemplo 1. pdf_findfont() example

```
<?php

$font = pdf_findfont($pdf, "Times New Roman", "winansi", 1);
if ($font) {
    pdf_setfont($pdf, $font, 10);
}

?>
```

pdf_get_buffer

(PHP 4 >= 4.0.5, PHP 5)

pdf_get_buffer -- Fetch the buffer containing the generated PDF data

Description

string **pdf_get_buffer** (resource pdfdoc)

Get the contents of the PDF output buffer. The result must be used by the client before calling any other PDFlib function.

pdf_get_font

pdf_get_font -- Deprecated: font handling

Description

This function is deprecated, use **pdf_get_value()** instead.

pdf_get_fontname

pdf_get_fontname -- Deprecated: font handling

Description

This function is deprecated, use **pdf_get_parameter()** instead.

pdf_get_fontsize

pdf_get_fontsize -- Deprecated: font handling

Description

This function is deprecated, use **pdf_get_value()** instead.

pdf_get_image_height

pdf_get_image_height -- Deprecated: returns height of an image

Description

This function is deprecated, use **pdf_get_value()** instead.

pdf_get_image_width

pdf_get_image_width -- Deprecated: Returns width of an image

Description

This function is deprecated, use **pdf_get_value()** instead.

pdf_get_majorversion

(PHP 4 >= 4.2.0, PHP 5)

pdf_get_majorversion -- Returns the major version number of the PDFlib

Description

int **pdf_get_majorversion** (void)

Returns the major version number of the PDFlib.

See also **pdf_get_minorversion()**.

pdf_get_minorversion

(PHP 4 >= 4.2.0, PHP 5)

pdf_get_minorversion -- Returns the minor version number of the PDFlib

Description

int **pdf_get_minorversion** (void)

Returns the minor version number of the PDFlib.

See also **pdf_get_majorversion()**.

PDF_get_parameter

(PHP 4 >= 4.0.1, PHP 5)

PDF_get_parameter -- Gets certain parameters

Description

string **pdf_get_parameter** (int pdf document, string name, double modifier)

The **PDF_get_parameter()** function gets several parameters of pdflib which are of the type string. The function parameter *modifier* characterizes the parameter to get. If the modifier is not needed it has to be 0.

See also **PDF_get_value()**, **PDF_set_value()**, **PDF_set_parameter()**.

pdf_get_pdi_parameter

(PHP 4 >= 4.0.5, PHP 5)

pdf_get_pdi_parameter -- Get some PDI string parameters

Description

string **pdf_get_pdi_parameter** (resource pdfdoc, string key, int document, int page, int index)

Get the contents of some PDI *document* parameter with string type.

pdf_get_pdi_value

(PHP 4 >= 4.0.5, PHP 5)

pdf_get_pdi_value -- Gets some PDI numerical parameters

Description

string **pdf_get_pdi_value** (resource pdfdoc, string key, int doc, int page, int index)

Get the contents of some PDI document parameter with numerical type.

PDF_get_value

(PHP 4 >= 4.0.1, PHP 5)

PDF_get_value -- Gets certain numerical value

Description

double **pdf_get_value** (int pdf document, string name, double modifier)

The **PDF_get_value()** function gets several numerical parameters of pdflib. The function parameter *modifier* characterizes the parameter to get. If the modifier is not needed it has to be 0.

See also **PDF_set_value()**, **PDF_get_parameter()**, **PDF_set_parameter()**.

pdf_initgraphics

(PHP 4 >= 4.0.5, PHP 5)

pdf_initgraphics -- Resets graphic state

Description

bool **pdf_initgraphics** (resource pdfdoc)

Reset all implicit color and graphics state parameters to their defaults. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

PDF_lineto

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_lineto -- Draws a line

Description

void **pdf_lineto** (int pdf document, double x-coor, double y-coor)

The **PDF_lineto()** function draws a line from the current point to the point with coordinates (*x-coor*, *y-coor*).

See also **PDF_moveto()**, **PDF_curveto()**, **PDF_stroke()**.

pdf_makespotcolor

(PHP 4 >= 4.0.5, PHP 5)

pdf_makespotcolor -- Makes a spotcolor

Description

bool **pdf_makespotcolor** (resource pdfdoc, string spotname)

Make a named spot color from the current color. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

See also **pdf_setcolor()**.

PDF_moveto

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_moveto -- Sets current point

Description

void **pdf_moveto** (int pdf document, double x-coor, double y-coor)

The **PDF_moveto()** function sets the current point to the coordinates *x-coor* and *y-coor*.

pdf_new

(PHP 4 >= 4.0.5, PHP 5)

pdf_new -- Creates a new pdf resource

Description

resource **pdf_new** ( )

Create a new PDF resource, using default error handling and memory management.

See also **pdf_close()**.

pdf_open_ccitt

(PHP 4 >= 4.0.5, PHP 5)

pdf_open_ccitt -- Opens a new image file with raw CCITT data

Description

int **pdf_open_ccitt** (resource pdfdoc, string filename, int width, int height, int BitReverse, int k, int Blackls1)

Open a raw CCITT image.

pdf_open_file

(PHP 4 >= 4.0.5, PHP 5)

pdf_open_file -- Opens a new pdf object

Description

bool **pdf_open_file** (resource pdfdoc, string filename)

Create a new PDF file using the supplied file name. If *filename* is empty the PDF document will be generated in memory instead of on file. The result must be fetched by the client with the **pdf_get_buffer()** function. Parameter *filename* was optional before PHP 5. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

The following example shows how to create a pdf document in memory and how to output it correctly.

Ejemplo 1. Creating a PDF document in memory

```
<?php

$pdf = pdf_new();

pdf_open_file($pdf);
pdf_begin_page($pdf, 595, 842);
pdf_set_font($pdf, "Times-Roman", 30, "host");
pdf_set_value($pdf, "textrendering", 1);
pdf_show_xy($pdf, "A PDF document created in memory!", 50, 750);
pdf_end_page($pdf);
pdf_close($pdf);

$data = pdf_get_buffer($pdf);

header("Content-type: application/pdf");
header("Content-disposition: inline; filename=test.pdf");
header("Content-length: " . strlen($data));

echo $data;

?>
```

PDF_open_gif

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

PDF_open_gif -- Opens a GIF image

Description

int **pdf_open_gif** (int pdf document, string filename)

The **PDF_open_gif()** function opens an image stored in the file with the name *filename*. The format of the image has to be gif. The function returns a pdf image identifier.

Ejemplo 1. Including a gif image

```
<?php
$im = PDF_open_gif($pdf, "test.gif");
pdf_place_image($pdf, $im, 100, 100, 1);
pdf_close_image($pdf, $im);
?>
```

See also **PDF_close_image()**, **PDF_open_jpeg()**, **PDF_open_memory_image()**, **PDF_execute_image()**, **PDF_place_image()**, **PDF_put_image()**.

pdf_open_image_file

(PHP 3 CVS only, PHP 4 , PHP 5)

pdf_open_image_file -- Reads an image from a file

Description

int **pdf_open_image_file** (resource pdfdoc, string imagetype, string filename, string stringparam, int intparam)

Open an image file. Supported types are *jpeg*, *tiff*, *gif*, and *png*. *stringparam* is either empty, *mask*, *masked*, or *page*. *intparam* is either 0, the image id of the applied mask, or the page. Parameters *stringparam* and *intparam* were optional before PHP 5.

pdf_open_image

(PHP 4 >= 4.0.5, PHP 5)

pdf_open_image -- Versatile function for images

Description

int **pdf_open_image** (resource pdfdoc, string imagetype, string source, string data, int length, int width, int height, int components, int bpc, string params)

Use image data from a variety of data sources. Supported types are *jpeg*, *ccitt*, *raw*. Supported sources are *memory*, *fileref*, *url*. *len* is only used when type is *raw*, *params* is only used when type is *ccitt*.

PDF_open_jpeg

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

PDF_open_jpeg -- Opens a JPEG image

Description

int **pdf_open_jpeg** (int pdf document, string filename)

The **PDF_open_jpeg()** function opens an image stored in the file with the name *filename*. The format of the image has to be jpeg. The function returns a pdf image identifier.

See also **PDF_close_image()**, **PDF_open_gif()**, **PDF_open_png()**, **PDF_open_memory_image()**, **PDF_execute_image()**, **PDF_place_image()**, **PDF_put_image()**.

PDF_open_memory_image

(PHP 3>= 3.0.10, PHP 4 , PHP 5)

PDF_open_memory_image -- Opens an image created with PHP's image functions

Description

int **pdf_open_memory_image** (int pdf document, int image)

The **PDF_open_memory_image()** function takes an image created with the PHP's image functions and makes it available for the pdf document. The function returns a pdf image identifier.

Ejemplo 1. Including a memory image

```
<?php
$im = ImageCreate(100, 100);
$col = ImageColorAllocate($im, 80, 45, 190);
ImageFill($im, 10, 10, $col);
$pim = PDF_open_memory_image($pdf, $im);
ImageDestroy($im);
pdf_place_image($pdf, $pim, 100, 100, 1);
pdf_close_image($pdf, $pim);
?>
```

See also **PDF_close_image()**, **PDF_open_jpeg()**, **PDF_open_gif()**, **PDF_open_png()** **PDF_execute_image()**, **PDF_place_image()**, **PDF_put_image()**.

pdf_open_pdi_page

(PHP 4 >= 4.0.5, PHP 5)

pdf_open_pdi_page -- Prepare a page

Description

int **pdf_open_pdi_page** (resource pdfdoc, int dochandle, int pagenumber, string pagelabel)

Prepare a page for later use with **pdf_place_image()**

pdf_open_pdi

(PHP 4 >= 4.0.5, PHP 5)

pdf_open_pdi -- Opens a PDF file

Description

int **pdf_open_pdi** (resource pdfdoc, string filename, string stringparam, int intparam)

Opens an existing PDF document and prepares it for later use.

See also **pdf_close_pdi()**.

PDF_open_png

(PHP 4 , PHP 5)

PDF_open_png -- Opens a PNG image

Description

int **pdf_open_png** (int pdf, string png_file)

The **PDF_open_png()** function opens an image stored in the file with the name *filename*. The format of the image has to be png. The function returns a pdf image identifier.

Ejemplo 1. Including a PNG image

```
<?php
$im = PDF_open_png ($pdf, "test.png");
pdf_place_image ($pdf, $im, 100, 100, 1);
pdf_close_image ($pdf, $im);
?>
```

See also **PDF_close_image()**, **PDF_open_jpeg()**, **PDF_open_gif()**, **PDF_open_memory_image()**, **PDF_execute_image()**, **PDF_place_image()**, **PDF_put_image()**.

pdf_open_tiff

pdf_open_tiff -- Deprecated: Opens a TIFF image

Description

This function is deprecated, use **pdf_open_image()** instead.

PDF_open

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_open -- Opens a new pdf document

Description

int **pdf_open** (int file, int info)

The **PDF_open()** function opens a new pdf document. The corresponding file has to be opened with **fopen()** and the file descriptor passed as argument *file*. *info* is the info structure that has to be created with **pdf_get_info()**. The info structure will be deleted within this function.

> **Nota:** The return value is needed as the first parameter in all other functions writing to the pdf document.
>
> **Nota:** This function does not allow the second parameter if pdflib 2.0 support is activated.

See also **fopen()**, **PDF_get_info()**, **PDF_close()**.

PDF_place_image

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

PDF_place_image -- Places an image on the page

Description

void **pdf_place_image** (int pdf document, int image, double x-coor, double y-coor, double scale)

The **PDF_place_image()** function places an image on the page at postion (*x-coor*, *x-coor*). The image can be scaled at the same time.

See also **PDF_put_image()**.

pdf_place_pdi_page

(PHP 4 >= 4.0.6, PHP 5)

pdf_place_pdi_page -- Places an image on the page

Description

bool **pdf_place_pdi_page** (resource pdfdoc, int page, float x, float y, float sx, float sy)

Place a PDI page with the lower left corner at (*x*, *y*), and scale it. Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

PDF_rect

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_rect -- Draws a rectangle

Description

void **pdf_rect** (int pdf document, double x-coor, double y-coor, double width, double height)

The **PDF_rect()** function draws a rectangle with its lower left corner at point (*x-coor*, *y-coor*). This width is set to *widgth*. This height is set to *height*.

See also **PDF_stroke()**.

PDF_restore

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_restore -- Restores formerly saved environment

Description

void **pdf_restore** (int pdf document)

The **PDF_restore()** function restores the environment saved with **PDF_save()**. It works like the postscript command grestore.

Ejemplo 1. Save and Restore

```
<?php PDF_save($pdf);
// do all kinds of rotations, transformations, ...
PDF_restore($pdf) ?>
```

See also **PDF_save()**.

PDF_rotate

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_rotate -- Sets rotation

Description

void **pdf_rotate** (int pdf document, double angle)

The **PDF_rotate()** function sets the rotation in degress to *angle*.

PDF_save

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_save -- Saves the current environment

Description

void **pdf_save** (int pdf document)

The **PDF_save()** function saves the current environment. It works like the postscript command gsave. Very useful if you want to translate or rotate an object without effecting other objects. **PDF_save()** should always be followed by **PDF_restore()** to restore the environment before **PDF_save()**.

See also **PDF_restore()**.

PDF_scale

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_scale -- Sets scaling

Description

void **pdf_scale** (int pdf document, double x-scale, double y-scale)

The **PDF_scale()** function sets the scaling factor in both directions. The following example scales x and y direction by 72. The following line will therefore be drawn one inch in both directions.

Ejemplo 1. Scaling

```
<?php PDF_scale($pdf, 72.0, 72.0);
PDF_lineto($pdf, 1, 1);
PDF_stroke($pdf);
?>
```

PDF_set_border_color

(PHP 3>= 3.0.12, PHP 4 , PHP 5)

PDF_set_border_color -- Sets color of border around links and annotations

Description

void **pdf_set_border_color** (int pdf document, double red, double green, double blue)

The **PDF_set_border_color()** function sets the color of the suroundig box of links and annotations. The three color components have to have a value between 0.0 and 1.0.

See also **PDF_set_border_style()**, **PDF_set_border_dash()**.

PDF_set_border_dash

(PHP 4 >= 4.0.1, PHP 5)

PDF_set_border_dash -- Sets dash style of border around links and annotations

Description

void **pdf_set_border_dash** (int pdf document, double black, double white)

The **PDF_set_border_dash()** function sets the lenght of black and white areas of a dashed line of the suroundig box of links and annotations.

See also **PDF_set_border_style()**, **PDF_set_border_color()**.

PDF_set_border_style

(PHP 3>= 3.0.12, PHP 4 , PHP 5)

PDF_set_border_style -- Sets style of border around links and annotations

Description

void **pdf_set_border_style** (int pdf document, string style, double width)

The **PDF_set_border_style()** function sets the style and width of the suroundig box of links and annotations. The parameter *style* can be 'solid' or 'dashed'.

See also **PDF_set_border_color()**, **PDF_set_border_dash()**.

PDF_set_char_spacing

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_set_char_spacing -- Sets character spacing

Description

void **pdf_set_char_spacing** (int pdf document, double space)

The **PDF_set_char_spacing()** function sets the spacing between characters.

See also **PDF_set_word_spacing()**, **PDF_set_leading()**.

PDF_set_duration

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_set_duration -- Sets duration between pages

Description

void **pdf_set_duration** (int pdf document, double duration)

The **PDF_set_duration()** function set the duration between following pages in seconds.

See also **PDF_set_transition()**.

PDF_set_font

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_set_font -- Selects a font face and size

Description

void **pdf_set_font** (int pdf document, string font name, double size, string encoding [, int embed])

The **PDF_set_font()** function sets the current font face, font size and encoding. If you use pdflib 0.6 you will need to provide the Adobe Font Metrics (afm-files) for the font in the font path (default is ./ fonts). If you use php3 or a version of pdflib older than 2.20 the fourth parameter *encoding* can take the following values: 0 = builtin, 1 = pdfdoc, 2 = macroman, 3 = macexpert, 4 = winansi. An encoding greater than 4 and less than 0 will default to winansi. winansi is often a good choice. If you use php4 and a version of pdflib >= 2.20 the encoding parameter has changed to a string. Use 'winansi', 'builtin', 'host', 'macroman' etc. instead. If the last parameter is set to 1 the font is embedded into the pdf document otherwise it is not. To embed a font is usually a good idea if the font is not widely spread and you cannot ensure that the person watching your document has access on fonts in the document. I font is only embedded once even if you call **PDF_set_font()** several times.

Nota: This function has to be called after **PDF_begin_page()** in order to create a valid pdf document.

Nota: If you reference a font in a .upr file make sure the name in the afm file and the font name are the same. Otherwise, the font will be embedded several times (Thanks to Paul Haddon for finding this.)

PDF_set_horiz_scaling

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_set_horiz_scaling -- Sets horizontal scaling of text

Description

void **pdf_set_horiz_scaling** (int pdf document, double scale)

The **PDF_set_horiz_scaling()** function sets the horizontal scaling to *scale* percent.

pdf_set_info_author

pdf_set_info_author -- Deprecated: Fills the author field of the document

Description

This function is deprecated, use **pdf_set_info()** instead.

pdf_set_info_creator

pdf_set_info_creator -- Deprecated: Fills the creator field of the document

Description

This function is deprecated, use **pdf_set_info()** instead.

pdf_set_info_keywords

pdf_set_info_keywords -- Deprecated: Fills the keywords field of the document

Description

This function is deprecated, use **pdf_set_info()** instead.

pdf_set_info_subject

pdf_set_info_subject -- Deprecated: Fills the subject field of the document

Description

This function is deprecated, use **pdf_set_info()** instead.

pdf_set_info_title

pdf_set_info_title -- Deprecated: Fills the title field of the document

Description

This function is deprecated, use **pdf_set_info()** instead.

PDF_set_info

(PHP 4 >= 4.0.1, PHP 5)

PDF_set_info -- Fills a field of the document information

Description

void **pdf_set_info** (int pdf document, string fieldname, string value)

The **PDF_set_info()** function sets an information field of a pdf document. Possible values for the fieldname are 'Subject', 'Title', 'Creator', 'Author', 'Keywords' and one user-defined name. It can be called before beginning a page.

Ejemplo 1. Setting document information

```
<?php
$fd = fopen("test.pdf", "w");
$pdfdoc = pdf_open($fd);
pdf_set_info($pdfdoc, "Author", "Uwe Steinmann");
pdf_set_info($pdfdoc, "Creator", "Uwe Steinmann");
pdf_set_info($pdfdoc, "Title", "Testing Info Fields");
pdf_set_info($pdfdoc, "Subject", "Test");
pdf_set_info($pdfdoc, "Keywords", "Test, Fields");
pdf_set_info($pdfdoc, "CustomField", "What ever makes sense");
pdf_begin_page($pdfdoc, 595, 842);
pdf_end_page($pdfdoc);
pdf_close($pdfdoc);
?>
```

> **Nota:** This function replaces **PDF_set_info_keywords()**, **PDF_set_info_title()**, **PDF_set_info_subject()**, **PDF_set_info_creator()**, **PDF_set_info_sybject()**.

PDF_set_leading

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_set_leading -- Sets distance between text lines

Description

void **pdf_set_leading** (int pdf document, double distance)

The **PDF_set_leading()** function sets the distance between text lines. This will be used if text is output by **PDF_continue_text()**.

See also **PDF_continue_text()**.

PDF_set_parameter

(PHP 4 , PHP 5)

PDF_set_parameter -- Sets certain parameters

Description

void **pdf_set_parameter** (int pdf document, string name, string value)

The **PDF_set_parameter()** function sets several parameters of pdflib which are of the type string.

See also **PDF_get_value()**, **PDF_set_value()**, **PDF_get_parameter()**.

PDF_set_text_matrix

(PHP 3>= 3.0.6)

PDF_set_text_matrix -- Sets the text matrix

Description

void **pdf_set_text_matrix** (int pdf document, array matrix)

The **PDF_set_text_matrix()** function sets a matrix which describes a transformation applied on the current text font. The matrix has to passed as an array with six elements.

PDF_set_text_pos

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_set_text_pos -- Sets text position

Description

void **pdf_set_text_pos** (int pdf document, double x-coor, double y-coor)

The **PDF_set_text_pos()** function sets the position of text for the next **pdf_show()** function call.

See also **PDF_show()**, **PDF_show_xy()**.

PDF_set_text_rendering

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_set_text_rendering -- Determines how text is rendered

Description

void **pdf_set_text_rendering** (int pdf document, int mode)

The **PDF_set_text_rendering()** function determines how text is rendered. The possible values for *mode* are 0=fill text, 1=stroke text, 2=fill and stroke text, 3=invisible, 4=fill text and add it to cliping path, 5=stroke text and add it to clipping path, 6=fill and stroke text and add it to cliping path, 7=add it to clipping path.

PDF_set_text_rise

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_set_text_rise -- Sets the text rise

Description

void **pdf_set_text_rise** (int pdf document, double rise)

The **PDF_set_text_rise()** function sets the text rising to *rise* points.

PDF_set_value

(PHP 4 >= 4.0.1, PHP 5)

PDF_set_value -- Sets certain numerical value

Description

void **pdf_set_value** (int pdf document, string name, double value)

The **PDF_set_value()** function sets several numerical parameters of pdflib.

See also **PDF_get_value()**, **PDF_get_parameter()**, **PDF_set_parameter()**.

PDF_set_word_spacing

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_set_word_spacing -- Sets spacing between words

Description

void **pdf_set_word_spacing** (int pdf document, double space)

The **PDF_set_word_spacing()** function sets the spacing between words.

See also **PDF_set_char_spacing()**, **PDF_set_leading()**.

pdf_setcolor

(PHP 4 >= 4.0.5, PHP 5)

pdf_setcolor -- Sets fill and stroke color

Description

bool **pdf_setcolor** (resource pdfdoc, string type, string colorspace, float c1, float c2, float c3, float c4)

Set the current color space and color. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo. The parameter *type* can be *fill*, *stroke* or *both* to specify that the color is set for filling, stroking or both filling and stroking. The parameter *colorspace* can be *gray*, *rgb*, *cmyk*, *spot* or *pattern*. The parameters *c1*, *c2*, *c3* and *c4* represent the color components for the color space specified by *colorspace*. Except as otherwise noted, the color components are floating-point values that range from 0 to 1. Parameters *c2*, *c3* and *c4* were optional before PHP 5.

For *gray* only *c1* is used.

For *rgb* parameters *c1*, *c2*, and *c3* specify the red, green and blue values respectively.

```
<?php
// Set fill and stroke colors to white.
pdf_setcolor($pdf, "both", "rgb", 1, 1, 1);
?>
```

For *cmyk*, parameters *c1*, *c2*, *c3*, and *c4* are the cyan, magenta, yellow and black values, respectively.

```
<?php
// Set fill and stroke colors to black.
pdf_setcolor($pdf, "both", "cmyk", 0, 0, 0, 1);
?>
```

For *spot*, *c1* should be a spot color handles returned by **pdf_makespotcolor()** and *c2* is a tint value between 0 and 1.

For *pattern*, *c1* should be a pattern handle returned by **pdf_begin_pattern()**.

PDF_setdash

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_setdash -- Sets dash pattern

Description

void **pdf_setdash** (int pdf document, double white, double black)

The **PDF_setdash()** function sets the dash pattern *white* white points and *black* black points. If both are 0 a solid line is set.

PDF_setflat

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_setflat -- Sets flatness

Description

void **pdf_setflat** (int pdf document, double value)

The **PDF_setflat()** function sets the flatness to a value between 0 and 100.

pdf_setfont

(PHP 4 >= 4.0.5, PHP 5)

pdf_setfont -- Set the current font

Description

bool **pdf_setfont** (resource pdfdoc, int font, float size)

Set the current font in the given *size*, using a *font* handle returned by **pdf_findfont()**. Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

See also **pdf_findfont()**.

PDF_setgray_fill

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_setgray_fill -- Sets filling color to gray value

Description

void **pdf_setgray_fill** (int pdf document, double gray value)

The **PDF_setgray_fill()** function sets the current gray value to fill a path.

See also **PDF_setrgbcolor_fill()**.

PDF_setgray_stroke

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_setgray_stroke -- Sets drawing color to gray value

Description

void **pdf_setgray_stroke** (int pdf document, double gray value)

The **PDF_setgray_stroke()** function sets the current drawing color to the given gray value.

See also **PDF_setrgbcolor_stroke()**.

PDF_setgray

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_setgray -- Sets drawing and filling color to gray value

Description

void **pdf_setgray** (int pdf document, double gray value)

The **PDF_setgray()** function sets the current drawing and filling color to the given gray value.

See also **PDF_setrgbcolor_stroke()**, **PDF_setrgbcolor_fill()**.

PDF_setlinecap

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_setlinecap -- Sets linecap parameter

Description

void **pdf_setlinecap** (int pdf document, int value)

The **PDF_setlinecap()** function sets the linecap parameter between a value of 0 and 2.

PDF_setlinejoin

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_setlinejoin -- Sets linejoin parameter

Description

void **pdf_setlinejoin** (int pdf document, long value)

The **PDF_setlinejoin()** function sets the linejoin parameter between a value of 0 and 2.

PDF_setlinewidth

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_setlinewidth -- Sets line width

Description

void **pdf_setlinewidth** (int pdf document, double width)

The **PDF_setlinewidth()** function sets the line width to *width*.

pdf_setmatrix

(PHP 4 >= 4.0.5, PHP 5)

pdf_setmatrix -- Sets current transformation matrix

Description

bool **pdf_setmatrix** (resource pdfdoc, float a, float b, float c, float d, float e, float f)

Explicitly set the current transformation matrix. Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

PDF_setmiterlimit

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_setmiterlimit -- Sets miter limit

Description

void **pdf_setmiterlimit** (int pdf document, double value)

The **PDF_setmiterlimit()** function sets the miter limit to a value greater of equal than 1.

pdf_setpolydash

pdf_setpolydash -- Deprecated: Sets complicated dash pattern

Description

This function is deprecated, use **pdf_setdash()** instead.

PDF_setrgbcolor_fill

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_setrgbcolor_fill -- Sets filling color to rgb color value

Description

void **pdf_setrgbcolor_fill** (int pdf document, double red value, double green value, double blue value)

The **PDF_setrgbcolor_fill()** function sets the current rgb color value to fill a path.

See also **PDF_setrgbcolor_fill()**.

PDF_setrgbcolor_stroke

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_setrgbcolor_stroke -- Sets drawing color to rgb color value

Description

void **pdf_setrgbcolor_stroke** (int pdf document, double red value, double green value, double blue value)

The **PDF_setrgbcolor_stroke()** function sets the current drawing color to the given rgb color value.

See also **PDF_setrgbcolor_stroke()**.

PDF_setrgbcolor

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_setrgbcolor -- Sets drawing and filling color to rgb color value

Description

void **pdf_setrgbcolor** (int pdf document, double red value, double green value, double blue value)

The **PDF_setrgbcolor_stroke()** function sets the current drawing and filling color to the given rgb color value.

See also **PDF_setrgbcolor_stroke()**, **PDF_setrgbcolor_fill()**.

PDF_show_boxed

(PHP 4 , PHP 5)

PDF_show_boxed -- Output text in a box

Description

int **pdf_show_boxed** (int pdf document, string text, double x-coor, double y-coor, double width, double height, string mode)

The **PDF_show_boxed()** function outputs the string *text* in a box with its lower left position at (*x-coor*, *y-coor*). The boxes dimension is *height* by *width*. The parameter *mode* determines how the text is type set. If *width* and *height* are zero, the *mode* can be "left", "right" or "center". If *width* or *height* is unequal zero it can also be "justify" and "fulljustify".

Returns the number of characters that could not be processed because they did not fit into the box.

See also **PDF_show()**, **PDF_show_xy()**.

PDF_show_xy

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_show_xy -- Output text at given position

Description

void **pdf_show_xy** (int pdf document, string text, double x-coor, double y-coor)

The **PDF_show_xy()** function outputs the string *text* at position (*x-coor*, *y-coor*).

See also **PDF_show()**.

PDF_show

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_show -- Output text at current position

Description

void **pdf_show** (int pdf document, string text)

The **PDF_show()** function outputs the string *text* at the current position using the current font.

See also **PDF_show_xy()**, **PDF_set_text_pos()**, **PDF_set_font()**.

PDF_skew

(PHP 4 , PHP 5)

PDF_skew -- Skews the coordinate system

Description

void **pdf_skew** (int pdf document, double alpha, double beta)

The **PDF_skew()** function skew the coordinate system by *alpha* (x) and *beta* (y) degrees. *alpha* and *beta* may not be 90 or 270 degrees.

PDF_stringwidth

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_stringwidth -- Returns width of text using current font

Description

double **pdf_stringwidth** (int pdf document, string text)

The **PDF_stringwidth()** function returns the width of the string in *text* by using the current font. It requires a font to be set before with **PDF_set_font()**.

See also **PDF_set_font()**.

PDF_stroke

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_stroke -- Draws line along path

Description

void **pdf_stroke** (int pdf document)

The **PDF_stroke()** function draws a line along current path. The current path is the sum of all line drawing. Without this function the line would not be drawn.

See also **PDF_closepath()**, **PDF_closepath_stroke()**.

PDF_translate

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

PDF_translate -- Sets origin of coordinate system

Description

void **pdf_translate** (int pdf document, double x-coor, double y-coor)

The **PDF_translate()** function sets the origin of coordinate system to the point (*x-coor*, *y-coor*) relativ the current origin. The following example draws a line from (0, 0) to (200, 200) relative to the initial coordinate system. You have to set the current point after **PDF_translate()** and before you start drawing more objects.

Ejemplo 1. Translation

```
<?php PDF_moveto($pdf, 0, 0);
PDF_lineto($pdf, 100, 100);
PDF_stroke($pdf);
PDF_translate($pdf, 100, 100);
PDF_moveto($pdf, 0, 0);
PDF_lineto($pdf, 100, 100);
PDF_stroke($pdf);
?>
```

XCVIII. PDO Functions

Introducción

Aviso
Esta extensión es *EXPERIMENTAL*. Esto significa que el comportamiento de esta extensión, los nombre de sus funciones y en definitiva TODO lo documentado sobre esta extensión, puede cambiar en una futura versión de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

The PHP Data Objects (PDO) extension defines a lightweight, consistent interface for accessing databases in PHP. Each database driver that implements the PDO interface can expose database-specific features as regular extension functions. Note that you cannot perform any database functions using the PDO extension by itself; you must use a database-specific PDO driver to access a database server.

Instalación

Windows

Follow the same steps to install and enable the PDO drivers of your choice.

1. Windows users can download the extension DLL `php_pdo.dll` as part of the PECL collection binaries from http://www.php.net/downloads.php or a more recent version from a PHP 5 PECL Snapshot.

2. To enable the PDO extension on Windows operating systems, you must add the following line to `php.ini`:
   ```
   extension=php_pdo.dll
   ```
3. Next, choose the other DB specific DLL files and either use **dl()** to load them at runtime, or enable them in `php.ini` below `pdo_pdo.dll`. For example:

```
extension=php_pdo.dll
extension=php_pdo_firebird.dll
extension=php_pdo_mssql.dll
extension=php_pdo_mysql.dll
extension=php_pdo_oci.dll
extension=php_pdo_oci8.dll
extension=php_pdo_odbc.dll
extension=php_pdo_pgsql.dll
extension=php_pdo_sqlite.dll
```

These DLL's should exist in the systems extension_dir.

Linux and UNIX

Due to a bug in the **pear** installer you should install the PDO package manually using the following steps:

Follow the same steps to install and enable the PDO drivers of your choice.

1. Download the PDO package to your local machine:
   ```
   bash$  wget http://pecl.php.net/get/PDO
   ```
2. Determine your PHP `bin` directory. If your PHP 5 CLI binary lives at `/usr/local/php5/bin/php` then the bin dir is `/usr/local/php5/bin`.
3. Set your path so that your PHP `bin` directory is at the front:
   ```
   export PATH="/usr/local/php5/bin:$PATH"
   ```
4. Manually build and install the PDO extension:
   ```
   bash$ tar xzf PDO-0.2.tgz
   bash$ cd PDO-0.2
   bash$ phpize
   bash$ ./configure
   bash$ make
   bash$ sudo -s
   bash# make install
   bash# echo extension=pdo.so >> /usr/local/php5/lib/php.ini
   ```

PDO Drivers

The following drivers currently implement the PDO interface:

Driver name	Supported databases
PDO_DBLIB	FreeTDS / Microsoft SQL Server / Sybase
PDO_FIREBIRD	Firebird/Interbase 6
PDO_MYSQL	MySQL 3.x/4.0
PDO_OCI	Oracle Call Interface
PDO_ODBC	ODBC v3 (IBM DB2 and unixODBC)
PDO_PGSQL	PostgreSQL
PDO_SQLITE	SQLite 3.x

Clases predefinidas

PDO

Represents a connection between PHP and a database server.

Constructor

- PDO - constructs a new PDO object

Métodos

- beginTransaction - begins a transaction
- commit - commits a transaction
- exec - issues an SQL statement and returns the number of affected rows
- errorCode - retrieves an error code, if any, from the database
- errorInfo - retrieves an array of error information, if any, from the database
- getAttribute - retrieves a database connection attribute
- lastInsertId - retrieves the value of the last row that was inserted into a table
- prepare - prepares an SQL statement for execution
- query - issues an SQL statement and returns a result set
- quote - returns a quoted version of a string for use in SQL statements
- rollBack - roll back a transaction
- setAttribute - sets a database connection attribute

PDOStatement

Represents a prepared statement and, after the statement is executed, an associated result set.

Métodos

- bindColumn - binds a PHP variable to an output column in a result set
- bindParam - binds a PHP variable to a parameter in the prepared statement
- columnCount - returns the number of columns in the result set

- errorCode - retrieves an error code, if any, from the statement
- errorInfo - retrieves an array of error information, if any, from the statement
- execute - executes a prepared statement
- fetch - fetches a row from a result set
- fetchAll - fetches an array containing all of the rows from a result set
- fetchSingle - returns the data from the first column in a result set
- getAttribute - retrieves a PDOStatement attribute
- getColumnMeta - retrieves metadata for a column in the result set
- rowCount - returns the number of rows that were affected by the execution of an SQL statement
- setAttribute - sets a PDOStatement attribute
- setFetchMode - sets the fetch mode for a PDOStatement

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

`PDO_PARAM_NULL` (**integer**)

Represents the SQL NULL data type.

`PDO_PARAM_INT` (**integer**)

Represents the SQL INTEGER data type.

`PDO_PARAM_STR` (**integer**)

Represents the SQL CHAR, VARCHAR, or other string data type.

`PDO_PARAM_LOB` (**integer**)

Represents the SQL large object data type.

`PDO_PARAM_STMT` (**integer**)

`PDO_FETCH_LAZY` (**integer**)

Specifies that the fetch method shall return each row as an object with variable names that correspond to the column names returned in the result set. PDO_FETCH_LAZY creates the

object variable names as they are accessed.

PDO_FETCH_ASSOC (integer)

Specifies that the fetch method shall return each row as an array indexed by column name as returned in the corresponding result set.

PDO_FETCH_NUM (integer)

Specifies that the fetch method shall return each row as an array indexed by column number as returned in the corresponding result set, starting at column 0.

PDO_FETCH_BOTH (integer)

Specifies that the fetch method shall return each row as an array indexed by both column name and number as returned in the corresponding result set, starting at column 0.

PDO_FETCH_OBJ (integer)

Specifies that the fetch method shall return each row as an object with property names that correspond to the column names returned in the result set.

PDO_FETCH_BOUND (integer)

Specifies that the fetch method shall return TRUE and assign the values of the columns in the result set to the PHP variables to which they were bound with the **PDOStatement::bindParam()** or **PDOStatement::bindColumn()** methods.

PDO_FETCH_COLUMN (integer)

Specifies that the fetch method shall return only a single requested column from the next row in the result set.

PDO_FETCH_CLASS (integer)

Specifies that the fetch method shall return a new instance of the requested class, mapping the columns to named properties in the class.

PDO_FETCH_INTO (integer)

Specifies that the fetch method shall update an existing instance of the requested class, mapping the columns to named properties in the class.

PDO_ATTR_AUTOCOMMIT (integer)

PDO_ATTR_PREFETCH (integer)

PDO_ATTR_TIMEOUT (integer)

PDO_ATTR_ERRMODE (integer)

PDO_ATTR_SERVER_VERSION (integer)

PDO_ATTR_CLIENT_VERSION (integer)

PDO_ATTR_SERVER_INFO (integer)

PDO_ATTR_CONNECTION_STATUS (integer)

PDO_ATTR_CASE (integer)

Force column names to a specific case specified by the PDO_CASE_* constants.

PDO_ATTR_CURSOR_NAME (integer)

PDO_ATTR_CURSOR (integer)

PDO_ATTR_ORACLE_NULLS (integer)

PDO_ATTR_PERSISTENT (integer)

PDO_ERRMODE_SILENT (integer)

PDO_ERRMODE_WARNING (integer)

PDO_ERRMODE_EXCEPTION (integer)

PDO_CASE_NATURAL (integer)

Leave column names as returned by the database driver.

PDO_CASE_LOWER (integer)

Force column names to lower case.

PDO_CASE_UPPER (integer)

Force column names to upper case.

PDO_FETCH_ORI_NEXT (integer)

Fetch the next row in the result set. Valid only for scrollable cursors.

PDO_FETCH_ORI_PRIOR (integer)

Fetch the previous row in the result set. Valid only for scrollable cursors.

PDO_FETCH_ORI_FIRST (integer)

Fetch the first row in the result set. Valid only for scrollable cursors.

PDO_FETCH_ORI_LAST (integer)

Fetch the last row in the result set. Valid only for scrollable cursors.

PDO_FETCH_ORI_ABS (integer)

Fetch the requested row by row number from the result set. Valid only for scrollable cursors.

PDO_FETCH_ORI_REL (integer)

Fetch the requested row by relative position from the current position of the cursor in the result set. Valid only for scrollable cursors.

PDO_CURSOR_FWDONLY (integer)

Create a PDOStatement object with a forward-only cursor. This may improve the performance of your application but restricts your PDOStatement object to fetching one row at a time from the result set in a forward direction.

PDO_CURSOR_SCROLL (integer)

Create a PDOStatement object with a scrollable cursor. Pass the PDO_FETCH_ORI_* constants to control the rows fetched from the result set.

PDO_ERR_CANT_MAP (integer)

PDO_ERR_SYNTAX (integer)

PDO_ERR_CONSTRAINT (integer)

PDO_ERR_NOT_FOUND (integer)

PDO_ERR_ALREADY_EXISTS (integer)

PDO_ERR_NOT_IMPLEMENTED (integer)

PDO_ERR_MISMATCH (integer)

PDO_ERR_TRUNCATED (integer)

PDO_ERR_DISCONNECTED (integer)

PDO_ERR_NO_PERM (integer)

PDO_ERR_NONE (string)

Corresponds to SQLSTATE '00000', meaning that the SQL statement was successfully issued with no errors or warnings.

Tabla de contenidos

PDO::prepare -- Prepares a statement for execution and returns a statement object
PDO::query -- Executes an SQL statement, returning a result set as a PDOStatement object
PDO::quote -- Quotes a string for use in a query.
PDO::rollBack -- Rolls back a transaction
PDO::setAttribute -- Set an attribute
PDOStatement::bindColumn -- Bind a column to a PHP variable
PDOStatement::bindParam -- Binds a parameter to a the specified variable name
PDOStatement::columnCount -- Returns the number of columns in the result set
PDOStatement::errorCode -- Fetch the SQLSTATE associated with the last operation on the statement handle
PDOStatement::errorInfo -- Fetch extended error information associated with the last operation on the statement handle
PDOStatement::execute -- Executes a prepared statement
PDOStatement::fetch -- Fetches the next row from a result set
PDOStatement::fetchAll -- Returns an array containing all of the result set rows
PDOStatement::fetchSingle -- Returns the first column in the next row of a result set
PDOStatement::getAttribute -- Retrieve a statement attribute
PDOStatement::getColumnMeta -- Returns meta data for a numbered column
PDOStatement::rowCount -- Returns the number of rows affected by the last SQL statement
PDOStatement::setAttribute -- Set a statement attribute
PDOStatement::setFetchMode -- Set the default fetch mode for this statement

PDO::beginTransaction

(no version information, might be only in CVS)

PDO::beginTransaction -- Initiates a transaction

Descripción

bool **PDO::beginTransaction** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Turns off autocommit mode. Call **PDO::commit()** or **PDO::rollback()** to end the transaction and return to autocommit mode.

Ejemplos

Ejemplo 1. Roll back a transaction

```
<?php
/* Begin a transaction, turning off autocommit */
$dbh->beginTransaction();

/* Change the database schema and data */
$sth = $dbh->exec("DROP TABLE fruit");
$sth = $dbh->exec("UPDATE dessert
    SET name = 'hamburger'");

/* Recognize mistake and roll back changes */
$dbh->rollBack();

/* Database connection is now back in autocommit mode */
?>
```

Ver también

PDO::commit()
PDO::rollBack()

PDO::commit

(no version information, might be only in CVS)

PDO::commit -- Commits a transaction

Descripción

bool **PDO::commit** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Commits a transaction, returning the database connection to autocommit mode until the next call to **PDO::beginTransaction()** starts a new transaction.

Ejemplos

Ejemplo 1. Commit a transaction

```
<?php
/* Begin a transaction, turning off autocommit */
$dbh->beginTransaction();

/* Change the database schema */
$sth = $dbh->exec("DROP TABLE fruit");

/* Commit the changes */
$dbh->commit();

/* Database connection is now back in autocommit mode */
?>
```

Ver también

PDO::beginTransaction()
PDO::rollBack()

PDO::__construct

(no version information, might be only in CVS)

PDO::__construct -- Creates a PDO instance representing a connection to a database

Descripción

PDO **PDO::__construct** (string dsn [, string username [, string password [, array driver_options]]])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Creates a PDO instance to represent a connection to the requested database.

Lista de parámetros

dsn

The Data Source Name, or DSN, contains the information required to connect to the database.

In general, a DSN consists of the PDO driver name, followed by a colon, followed by the PDO driver-specific connection syntax. Examples of each driver are given below:

PDO_DBLIB

The DSN prefix is either `sybase:` or `mssql:` depending on which libraries it was linked against during compilation.

```
sybase:host=localhost; dbname=testdb
```

```
mssql:host=localhost; dbname=testdb
```

PDO_FIREBIRD

```
firebird:User=john;Password=mypass;Database=DATABASE.GDE;Da
taSource=localhost;Port=3050
```

PDO_MYSQL

```
mysql:host=localhost;dbname=testdb
```

PDO_OCI

To connect via `tnsnames.ora`, use:

```
oci:mydb
```

If using instantclient, use:

```
oci:dbname=//localhost:1521/testdb
```

PDO_ODBC

```
odbc:DSN=SAMPLE;UID=john;PWD=mypass
```

DSN=SAMPLE refers to the SAMPLE data source configured in the ODBC driver manager.

PDO_PGSQL

```
pgsql:host=localhost port=5432 dbname=testdb user=john
password=mypass
```

Note, by passing *user* and *password* in the DSN, the *username* and *password* parameters become optional. If specified, they are glued to the end of the connection string.

PDO_SQLITE

```
sqlite:/path/to/database
```

To create a database in memory, use:

```
sqlite::memory:
```

The *dsn* parameter supports three different methods of specifying the arguments required to create a database connection:

Driver invocation

dsn contains the full DSN.

URI invocation

dsn consists of `uri:` followed by a URI that defines the location of a file containing the DSN string. The URI can specify a local file or a remote URL.

```
uri:file:///path/to/dsnfile
```

Aliasing

dsn consists of a name *name* that maps to *pdo.dsn.name* in `php.ini` defining the DSN string.

> **Nota:** The alias must be defined in `php.ini`, and not `.htaccess` or `httpd.conf`

username

The user name for the DSN string. This parameter is optional for some PDO drivers.

password

The password for the DSN string. This parameter is optional for some PDO drivers.

driver_options

A key=>value array of driver-specific connection options.

Valores retornados

Returns a PDO object on success.

Exceptions

PDO::construct() throws a PDOException if the attempt to connect to the requested database fails.

Ejemplos

Ejemplo 1. Create a PDO instance via driver invocation

```
<?php
// Connect to an ODBC database using driver invocation
$dsn = 'mysql:dbname=testdb;host=127.0.0.1';
$user = 'dbuser';
$password = 'dbpass';

try {
    $dbh = new PDO($dsn, $user, $password);
} catch (PDOException $e) {
    echo 'Connection failed: ' . $e->getMessage();
}

?>
```

Ejemplo 2. Create a PDO instance via URI invocation

The following example assumes that the file `/usr/local/dbconnect` exists with file permissions that enable PHP to read the file. The file contains the PDO DSN to connect to a DB2 database through the PDO_ODBC driver:

```
odbc:DSN=SAMPLE;UID=john;PWD=mypass
```

The PHP script can then create a database connection by simply passing the *uri:* parameter and pointing to the file URI:

```
<?php
// Connect to an ODBC database using driver invocation
$dsn = 'uri:file:///usr/local/dbconnect';
$user = '';
$password = '';

try {
    $dbh = new PDO($dsn, $user, $password);
} catch (PDOException $e) {
    echo 'Connection failed: ' . $e->getMessage();
}

?>
```

Ejemplo 3. Create a PDO instance using an alias

The following example assumes that `php.ini` contains the following entry to enable a connection to a MySQL database using only the alias *mydb*:

```
[PDO]
pdo.dsn.mydb="mysql:dbname=testdb;host=localhost"
```

```
<?php
// Connect to an ODBC database using an alias
$dsn = 'mydb';
$user = '';
$password = '';

try {
    $dbh = new PDO($dsn, $user, $password);
}catch (PDOException $e) {
    echo 'Connection failed: ' . $e->getMessage();
}

?>
```

PDO::errorCode

(no version information, might be only in CVS)

PDO::errorCode -- Fetch the SQLSTATE associated with the last operation on the database handle

Descripción

int **PDO::errorCode** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Valores retornados

Returns a SQLSTATE, a five-character alphanumeric identifier defined in the ANSI SQL standard.

PDO::errorCode() only retrieves error codes for operations performed directly on the database handle. If you create a PDOStatement object through **PDO::prepare()** or **PDO::query()** and invoke an error on the statement handle, **PDO::errorCode()** will return *PDO_ERR_NONE*. You must call **PDOStatement::errorCode()** to return the error code for an operation performed on a particular statement handle.

Ejemplos

Ejemplo 1. Retrieving a SQLSTATE code

```
<?php
/* Provoke an error -- the BONES table does not exist */
$dbh->exec("INSERT INTO bones(skull) VALUES ('lucy')");

echo "\nPDO::errorCode(): ";
print $dbh->errorCode();
?>
```

El resultado del ejemplo seria:

```
PDO::errorCode(): 42S02
```

Ver también

PDO::errorInfo()
PDOStatement::errorCode()
PDOStatement::errorInfo()

PDO::errorInfo

(no version information, might be only in CVS)

PDO::errorInfo -- Fetch extended error information associated with the last operation on the database handle

Descripción

array **PDO::errorInfo** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Valores retornados

PDO::errorInfo() returns an array of error information about the last operation performed by this database handle. The array consists of the following fields:

Element	Information
0	SQLSTATE error code (a five-character alphanumeric identifier defined in the ANSI SQL standard).
1	Driver-specific error code.
2	Driver-specific error message.

PDO::errorInfo() only retrieves error information for operations performed directly on the database handle. If you create a PDOStatement object through **PDO::prepare()** or **PDO::query()** and invoke an error on the statement handle, **PDO::errorInfo()** will insert an error code of *PDO_ERR_NONE* into the first element of the returned array. You must call **PDOStatement::errorInfo()** to return the error information for an operation performed on a particular statement handle.

Ejemplos

Ejemplo 1. Displaying errorInfo() fields for a PDO_ODBC connection to a DB2 database

```
<?php
/* Provoke an error -- the BONES table does not exist */
$err = $dbh->prepare('SELECT skull FROM bones');
$err->execute();
echo "\nPDO::errorInfo():\n";
print_r($err->errorInfo());
?>
```

El resultado del ejemplo seria:

```
PDO::errorInfo():
Array
(
    [0] => 42S02
    [1] => -204
    [2] => [IBM][CLI Driver][DB2/LINUX] SQL0204N  "DANIELS.BONES" is an undefined name.
)
```

Ver también

PDO::errorCode()
PDOStatement::errorCode()
PDOStatement::errorInfo()

PDO::exec

(no version information, might be only in CVS)

PDO::exec -- Execute an SQL statement and return the number of affected rows

Descripción

long **PDO::exec** (string statement)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

PDO::exec() prepares and executes an SQL statement in a single function call, returning the number of rows affected by the statement.

PDO::exec() does not return results from a SELECT statement. For a SELECT statement that you only need to issue once during your program, consider issuing **PDO::query()**. For a SELECT statement that you need to issue multiple times, prepare a PDOStatement object with **PDO::prepare()** and issue the statement with **PDOStatement::execute()**.

Lista de parámetros

statement

The SQL statement to prepare and execute.

Valores retornados

PDO::exec() returns the number of rows that were modified or deleted by the SQL statement you issued. If no rows were affected, **PDO::exec()** returns *0*.

Ejemplos

Ejemplo 1. Issuing a DELETE statement

Count the number of rows deleted by a DELETE statement with no WHERE clause.

```
<?php
$dbh = new PDO('odbc:sample', 'db2inst1', 'ibmdb2');

/* Delete all rows from the FRUIT table */
$count = $dbh->exec("DELETE FROM fruit WHERE colour = 'red'");

/* Return number of rows that were deleted */
print("Return number of rows that were deleted:\n");
print("Deleted $count rows.\n");
?>
```

El resultado del ejemplo seria:

```
Return number of rows that were deleted:
Deleted 1 rows.
```

Ver también

PDO::prepare()
PDO::query()
PDOStatement::execute()

PDO::getAttribute

(no version information, might be only in CVS)

PDO::getAttribute -- Retrieve a database connection attribute

Descripción

mixed **PDO::getAttribute** (long attribute)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

This function returns the value of a database connection attribute. To retrieve PDOStatement attributes, refer to **PDOStatement::getAttribute()**.

Note that some databases may not support all of the database connection attributes.

Lista de parámetros

attribute

One of the *PDO_ATTR_** constants. The constants that apply to database connections are as follows:

PDO_ATTR_AUTOCOMMIT
PDO_ATTR_CASE
PDO_ATTR_CLIENT_VERSION
PDO_ATTR_CONNECTION_STATUS
PDO_ATTR_ERRMODE
PDO_ATTR_ORACLE_NULLS
PDO_ATTR_PERSISTENT
PDO_ATTR_PREFETCH
PDO_ATTR_SERVER_INFO
PDO_ATTR_SERVER_VERSION
PDO_ATTR_TIMEOUT

Valores retornados

A successful call returns the value of the requested PDO attribute. An unsuccessful call returns *null*.

Ejemplos

Ejemplo 1. Retrieving database connection attributes

```php
<?php
$conn = new PDO('odbc:sample', 'db2inst1', 'ibmdb2');

print "\nPDO_ATTR_AUTOCOMMIT: ";
print $conn->getAttribute(PDO_ATTR_AUTOCOMMIT);

print "\nPDO_ATTR_ERRMODE: ";
print $conn->getAttribute(PDO_ATTR_ERRMODE);

print "\nPDO_ATTR_CASE: ";
print $conn->getAttribute(PDO_ATTR_CASE);

print "\nPDO_ATTR_CLIENT_VERSION: ";
print $conn->getAttribute(PDO_ATTR_CLIENT_VERSION);

print "\nPDO_ATTR_CONNECTION_STATUS: ";
print $conn->getAttribute(PDO_ATTR_CONNECTION_STATUS);

print "\nPDO_ATTR_ORACLE_NULLS: ";
print $conn->getAttribute(PDO_ATTR_ORACLE_NULLS);

print "\nPDO_ATTR_PERSISTENT: ";
print $conn->getAttribute(PDO_ATTR_PERSISTENT);

print "\nPDO_ATTR_PREFETCH: ";
print $conn->getAttribute(PDO_ATTR_PREFETCH);

print "\nPDO_ATTR_SERVER_INFO: ";
print $conn->getAttribute(PDO_ATTR_SERVER_INFO);

print "\nPDO_ATTR_SERVER_VERSION: ";
print $conn->getAttribute(PDO_ATTR_SERVER_VERSION);

print "\nPDO_ATTR_TIMEOUT: ";
print $conn->getAttribute(PDO_ATTR_TIMEOUT);
?>
```

Ver también

PDO::setAttribute()
PDOStatement::getAttribute()
PDOStatement::setAttribute()

PDO::lastInsertId

(no version information, might be only in CVS)

PDO::lastInsertId -- Returns the ID of the last inserted row

Description

int **PDO::lastInsertId** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Nota: Due to differences between database server implementations, this method may

not always return a meaningful result.

Valores retornados

Returns an integer representing the row ID of the last row that was inserted into the database. If the PDO driver does not support this capability, **PDO::lastInsertID()** issues a PDOWarning exception.

PDO::prepare

(no version information, might be only in CVS)

PDO::prepare -- Prepares a statement for execution and returns a statement object

Descripción

PDOStatement **PDO::prepare** (string statement [, array driver_options])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Prepares an SQL statement to be executed by the **PDOStatement::execute()** method. The SQL statement can contain zero or more named (:name) or question mark (?) parameter markers for which real values will be substituted when the statement is executed.

Calling **PDO::prepare()** and **PDOStatement::execute()** for statements that will be issued multiple times with different parameter values optimizes the performance of your application and helps prevent SQL injection attacks.

Lista de parámetros

statement

This must be a valid SQL statement for the target database server.

driver_options

This array holds one or more key=>value pairs to set attribute values for the PDOStatement object that this method returns. You would most commonly use this to set the *PDO_ATTR_CURSOR* value to *PDO_CURSOR_SCROLL* to request a scrollable cursor.

Valores retornados

If the database server successfully prepares the statement, **PDO::prepare()** returns a PDOStatement object.

Ejemplos

Ejemplo 1. Prepare an SQL statement with named parameters

```
<?php
/* Execute a prepared statement by passing an array of values */
$sql = 'SELECT name, colour, calories
    FROM fruit
    WHERE calories < :calories AND colour = :colour'
$sth = $dbh->prepare($sql, array(PDO_ATTR_CURSOR, PDO_CURSOR_FWDONLY));
$sth->execute(array(':calories' => 150, ':colour' => 'red'));
$red = $sth->fetchAll();
$sth->execute(array(':calories' => 175, ':colour' => 'yellow'));
$yellow = $sth->fetchAll();
?>
```

Ejemplo 2. Prepare an SQL statement with question mark parameters

```
<?php
/* Execute a prepared statement by passing an array of values */
$sth = $dbh->prepare('SELECT name, colour, calories
    FROM fruit
    WHERE calories < ? AND colour = ?');
$sth->execute(array(150, 'red'));
$red = $sth->fetchAll();
$sth->execute(array(175, 'yellow'));
$yellow = $sth->fetchAll();
?>
```

Ver también

PDO::exec()
PDO::query()
PDOStatement::execute()

PDO::query

(no version information, might be only in CVS)

PDO::query -- Executes an SQL statement, returning a result set as a PDOStatement object

Descripción

object **PDO::query** (string statement)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

PDO::query() prepares and executes an SQL statement in a single function call, returning the result set (if any) returned by the statement as a PDOStatement object.

For a SELECT statement that you need to issue multiple times, prepare a PDOStatement object with **PDO::prepare()** and issue the statement with **PDOStatement::execute()**.

Lista de parámetros

statement

The SQL statement to prepare and execute.

Valores retornados

PDO::query() returns a PDOStatement object.

Ejemplos

Ejemplo 1. Demonstrate PDO::query

A nice feature of **PDO::query()** is that it enables you to iterate over the rowset returned by a successfully executed SELECT statement.

```
<?php
function getFruit($conn) {
    $sql = 'SELECT name, colour, calories FROM fruit ORDER BY name';
    foreach ($conn->query($sql) as $row) {
        print $row['NAME'] . "\t";
        print $row['COLOUR'] . "\t";
        print $row['CALORIES'] . "\n";
    }
}
?>
```

El resultado del ejemplo seria:

```
apple   red     150
banana  yellow  250
kiwi    brown   75
lemon   yellow  25
orange  orange  300
pear    green   150
watermelon      pink    90
```

Ver también

PDO::exec()
PDO::prepare()
PDOStatement::execute()

PDO::quote

(no version information, might be only in CVS)

PDO::quote -- Quotes a string for use in a query.

Descripción

string **PDO::quote** (string string [, int parameter_type])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

PDO::quote() places quotes around the input string and escapes and single quotes within the input

string. Quoting input strings has been a common means of attempting to prevent SQL injection attacks; however, an even safer approach is to use prepared statements with named parameters or placeholders for the input values.

Not all PDO drivers implement this method.

Lista de parámetros

string

The string to be quoted.

parameter_type

Provides a data type hint for drivers that have alternate quoting styles. The default value is PDO_PARAM_STR.

Valores retornados

Returns a quoted string that is theoretically safe to pass into an SQL statement.

Ejemplos

Ejemplo 1. Quoting a normal string

```
<?php
$conn = new PDO('sqlite:/home/lynn/music.sql3');

/* Simple string */
$string = 'Nice';
print "Unquoted string: $string\n";
print "Quoted string: " . $conn->quote($string) . "\n";
?>
```

El resultado del ejemplo seria:

```
Unquoted string: Nice
Quoted string: 'Nice'
```

Ejemplo 2. Quoting a dangerous string

```
<?php
$conn = new PDO('sqlite:/home/lynn/music.sql3');

/* Dangerous string */
$string = 'Naughty \' string';
print "Unquoted string: $string\n";
print "Quoted string:" . $conn->quote($string) . "\n";
?>
```

El resultado del ejemplo seria:

```
Unquoted string: Naughty ' string
Quoted string: 'Naughty '' string'
```

Ejemplo 3. Quoting a complex string

```
<?php
$conn = new PDO('sqlite:/home/lynn/music.sql3');

/* Complex string */
$string = "Co'mpl''ex \"st'\"ring";
print "Unquoted string: $string\n";
print "Quoted string: " . $conn->quote($string) . "\n";
?>
```

El resultado del ejemplo seria:

```
Unquoted string: Co'mpl''ex "st'"ring
Quoted string: 'Co''mpl''''ex "st'''"ring'
```

Ver también

PDO::prepare()
PDOStatement::execute()

PDO::rollBack

(no version information, might be only in CVS)

PDO::rollBack -- Rolls back a transaction

Descripción

bool **PDO::rollBack** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

When issued against databases that support transactions, **PDO::rollBack()** rolls back any work in progress and and returns the connection state to autocommit mode.

You must issue **PDO::beginTransaction()** to set the connection state to manual commit mode before issuing **PDO::rollBack()** has any effect.

Ejemplos

Ejemplo 1. Roll back a transaction

```
<?php
/* Begin a transaction, turning off autocommit */
$dbh->beginTransaction();

/* Change the database schema and data */
$sth = $dbh->exec("DROP TABLE fruit");
$sth = $dbh->exec("UPDATE dessert
    SET name = 'hamburger'");

/* Recognize mistake and roll back changes */
$dbh->rollBack();

/* Database connection is now back in autocommit mode */
?>
```

Ver también

PDO::beginTransaction()
PDO::commit()

PDO::setAttribute

(no version information, might be only in CVS)

PDO::setAttribute -- Set an attribute

Description

bool **PDO::setAttribute** (int attribute, mixed value)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Sets a database connection attribute. The generic PDO connection attributes include:

- *PDO_ATTR_CASE*: Force column names to a specific case.
    - *PDO_CASE_LOWER*: Force column names to lower case.
    - *PDO_CASE_NATURAL*: Leave column names as returned by the database driver.
    - *PDO_CASE_UPPER*: Force column names to upper case.

PDO drivers may define further driver-specific attributes.

PDOStatement::bindColumn

(no version information, might be only in CVS)

PDOStatement::bindColumn -- Bind a column to a PHP variable

Descripción

bool **PDOStatement::bindColumn** (mixed column, mixed &param [, int type [, int maxlen [, mixed driver_options]]])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

On each row fetch *param* will contain the value of the corresponding column. *column* is the 1-based offset of the column, or the column name. For maximum portability, do not call this function before calling **PDOStatement::execute()**.

Lista de parámetros

column

Number of the column (1-indexed) in the result set.

param

Name of the PHP variable to which the column will be bound.

type

Data type of the parameter, specified by the PDO_PARAM_* constants.

maxlen

Maximum length of the parameter.

driver_options

Ejemplos

Ejemplo 1. Binding result set output to PHP variables

Binding columns in the result set to PHP variables is an effective way to make the data contained in each row immediately available to your application. The following example demonstrates how PDO allows you to bind and retrieve columns with a variety of options and with intelligent defaults.

```
<?php
function readData($dbh) {
  $sql = 'SELECT name, colour, calories FROM fruit';
  try {
    $stmt = $dbh->prepare($sql);
    $stmt->execute();

    /* Bind by column number with an explicit data type & length */
    $stmt->bindColumn(1, $name, PDO_PARAM_STR, 64);

    /* Bind by column number with default data type & length */
    $stmt->bindColumn(2, $colour);

    /* Bind by column name with default data type & length */
    $stmt->bindColumn('CALORIES', $cals);

    while ($row = $stmt->fetch(PDO_FETCH_BOUND)) {
      $data = $name . "\t" . $colour . "\t" . $cals . "\n";
      print $data;
    }
  }
  catch (PDOException $e) {
    print $e->getMessage();
  }
}
readData($dbh);
?>
```

El resultado del ejemplo seria:

```
apple   red     150
banana  yellow  175
kiwi    green   75
orange  orange  150
mango   red     200
strawberry      red     25
```

Ver también

PDOStatement::execute()
PDOStatement::fetch()
PDOStatement::fetchAll()
PDOStatement::fetchSingle()

PDOStatement::bindParam

(no version information, might be only in CVS)

PDOStatement::bindParam -- Binds a parameter to a the specified variable name

Description

bool **PDOStatement::bindParam** (mixed parameter_name, mixed &variable [, int data_type [, int length]])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Binds an SQL statement parameter to the specified variable name. The SQL statement parameter can either be a named placeholder or a question mark placeholder.

Output parameters will set the value of the bound PHP variable to the value returned by the database when the SQL statement is executed. This enables you to call stored procedures with output or input/output parameters, for example, for databases that support such features.

For input-only variables, you can pass an array of input values to **PDOStatement::execute()** instead.

Ejemplo 1. Execute a prepared statement with named placeholders

```
<?php
/* Execute a prepared statement by binding PHP variables */
$calories = 150;
$colour = 'red';
$sth = $dbh->prepare('SELECT name, colour, calories
    FROM fruit
    WHERE calories < :calories AND colour = :colour');
$sth->bindParam(':calories', $calories, PDO_PARAM_INT);
$sth->bindParam(':colour', $colour, PDO_PARAM_STR, 12);
$sth->execute();
?>
```

Ejemplo 2. Execute a prepared statement with question mark placeholders

```php
<?php
/* Execute a prepared statement by binding PHP variables */
$calories = 150;
$colour = 'red';
$sth = $dbh->prepare('SELECT name, colour, calories
    FROM fruit
    WHERE calories < ? AND colour = ?');
$sth->bindParam(1, $calories, PDO_PARAM_INT);
$sth->bindParam(2, $colour, PDO_PARAM_STR, 12);
$sth->execute();
?>
```

PDOStatement::columnCount

(no version information, might be only in CVS)

PDOStatement::columnCount -- Returns the number of columns in the result set

Descripción

int **PDOStatement::columnCount** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Use **PDOStatement::columnCount()** to return the number of columns in the result set represented by the PDOStatement object.

If the PDOStatement object was returned from **PDO::query()**, the column count is immediately available.

If the PDOStatement object was returned from **PDO::prepare()**, an accurate column count will not be available until you invoke **PDOStatement::execute()**.

Valores retornados

Returns the number of columns in the result set represented by the PDOStatement object. If there is no result set, **PDOStatement::columnCount()** returns *0*.

Ejemplos

Ejemplo 1. Counting columns

This example demonstrates how **PDOStatement::columnCount()** operates with and without a result set.

```php
<?php
$dbh = new PDO('odbc:sample', 'db2inst1', 'ibmdb2');

$sth = $dbh->prepare("SELECT name, colour FROM fruit");

/* Count the number of columns in the (non-existent) result set */
$colcount = $sth->columnCount();
print("Before execute(), result set has $colcount columns (should be 0)\n");

$sth->execute();

/* Count the number of columns in the result set */
$colcount = $sth->columnCount();
print("After execute(), result set has $colcount columns (should be 2)\n");

?>
```

El resultado del ejemplo seria:

```
Before execute(), result set has 0 columns (should be 0)
After execute(), result set has 2 columns (should be 2)
```

Ver también

PDO::prepare()
PDOStatement::execute()
PDOStatement::rowCount()

PDOStatement::errorCode

(no version information, might be only in CVS)

PDOStatement::errorCode -- Fetch the SQLSTATE associated with the last operation on the statement handle

Descripción

int **PDOStatement::errorCode** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Valores retornados

Returns a SQLSTATE, a five-character alphanumeric identifier defined in the ANSI SQL standard. **PDOStatement::errorCode()** only retrieves error codes for operations performed with PDOStatement objects.

Ejemplos

Ejemplo 1. Retrieving a SQLSTATE code

```
<?php
/* Provoke an error -- the BONES table does not exist */
$err = $dbh->prepare('SELECT skull FROM bones');
$err->execute();

echo "\nPDOStatement::errorCode(): ";
print $err->errorCode();
?>
```

El resultado del ejemplo seria:

```
PDOStatement::errorCode(): 42S02
```

Ver también

PDO::errorCode()
PDO::errorInfo()
PDOStatement::errorInfo()

PDOStatement::errorInfo

(no version information, might be only in CVS)

PDOStatement::errorInfo -- Fetch extended error information associated with the last operation on the statement handle

Descripción

array **PDOStatement::errorInfo** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Valores retornados

PDOStatement::errorInfo() returns an array of error information about the last operation performed by this statement handle. The array consists of the following fields:

Element	Information
0	SQLSTATE error code (a five-character alphanumeric identifier defined in the ANSI SQL standard).
1	Driver-specific error code.
2	Driver-specific error message.

Ejemplos

Ejemplo 1. Displaying errorInfo() fields for a PDO_ODBC connection to a DB2 database

```
<?php
/* Provoke an error -- the BONES table does not exist */
$sth = $dbh->prepare('SELECT skull FROM bones');
$sth->execute();

echo "\nPDOStatement::errorInfo():\n";
$arr = $sth->errorInfo();
print_r($arr);
?>
```

El resultado del ejemplo seria:

```
PDOStatement::errorInfo():
Array
(
    [0] => 42S02
    [1] => -204
    [2] => [IBM][CLI Driver][DB2/LINUX] SQL0204N  "DANIELS.BONES" is an undefined name.
)
```

Ver también

PDO::errorCode()
PDO::errorInfo()
PDOStatement::errorCode()

PDOStatement::execute

(no version information, might be only in CVS)

PDOStatement::execute -- Executes a prepared statement

Description

bool **PDOStatement::execute** ([array input_parameters])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Execute the prepared statement. If the prepared statement included parameter markers, you must either:

- call **PDOStatement::bindParam()** to bind PHP variables to the parameter markers: bound variables pass their value as input and receive the output value, if any, of their associated parameter markers
- or pass an array of input-only parameter values

Ejemplo 1. Execute a prepared statement with bound variables

```php
<?php
/* Execute a prepared statement by binding PHP variables */
$calories = 150;
$colour = 'red';
$sth = $dbh->prepare('SELECT name, colour, calories
    FROM fruit
    WHERE calories < :calories AND colour = :colour');
$sth->bindParam(':calories', $calories, PDO_PARAM_INT);
$sth->bindParam(':colour', $colour, PDO_PARAM_STR, 12);
$sth->execute();
?>
```

Ejemplo 2. Execute a prepared statement with an array of insert values

```php
<?php
/* Execute a prepared statement by passing an array of insert values */
$calories = 150;
$colour = 'red';
$sth = $dbh->prepare('SELECT name, colour, calories
    FROM fruit
    WHERE calories < :calories AND colour = :colour');
$sth->bindParam(':calories', $calories, PDO_PARAM_INT);
$sth->bindParam(':colour', $colour, PDO_PARAM_STR, 12);
$sth->execute(array(':calories' => $calories, ':colour' => $colour));
?>
```

Ejemplo 3. Execute a prepared statement with question mark placeholders

```php
<?php
/* Execute a prepared statement by binding PHP variables */
$calories = 150;
$colour = 'red';
$sth = $dbh->prepare('SELECT name, colour, calories
    FROM fruit
    WHERE calories < ? AND colour = ?');
$sth->bindParam(1, $calories, PDO_PARAM_INT);
$sth->bindParam(2, $colour, PDO_PARAM_STR, 12);
$sth->execute();
?>
```

PDOStatement::fetch

(no version information, might be only in CVS)

PDOStatement::fetch -- Fetches the next row from a result set

Descripción

mixed **PDOStatement::fetch** ([int fetch_style [, int cursor_orientation [, int cursor_offset]]])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Fetches a row from a result set associated with a PDOStatement object.

Lista de parámetros

fetch_style

Controls how the next row will be returned to the caller. This value must be one of the *PDO_FETCH_** constants, defaulting to *PDO_FETCH_BOTH*.

- *PDO_FETCH_ASSOC*: returns an array indexed by column name as returned in your result set
- *PDO_FETCH_BOTH* (default): returns an array indexed by both column name and column number as returned in your result set
- *PDO_FETCH_BOUND*: returns **`TRUE`** and assigns the values of the columns in your result set to the PHP variables to which they were bound with the **PDOStatement::bindParam()** method
- *PDO_FETCH_LAZY*: combines *PDO_FETCH_BOTH* and *PDO_FETCH_OBJ*, creating the object variable names as they are accessed
- *PDO_FETCH_OBJ*: returns an anonymous object with property names that correspond to the column names returned in your result set
- *PDO_FETCH_NUM*: returns an array indexed by column number as returned in your result set, starting at column 0

cursor_orientation

For a PDOStatement object representing a scrollable cursor, this value determines which row will be returned to the caller. This value must be one of the *PDO_FETCH_ORI_** constants, defaulting to *PDO_FETCH_ORI_NEXT*.

offset

For a PDOStatement object representing a scrollable cursor for which the *cursor_orientation* parameter is set to *PDO_FETCH_ORI_ABS*, this value specifies the absolute number of the row in the result set that shall be fetched.

For a PDOStatement object representing a scrollable cursor for which the *cursor_orientation* parameter is set to *PDO_FETCH_ORI_REL*, this value specifies the row to fetch relative to the cursor position before **PDOStatement::fetch()** was called.

Ejemplos

Ejemplo 1. Fetching rows using different fetch styles

```
<?php
$sth = $dbh->prepare("SELECT name, colour FROM fruit");
$sth->execute();

/* Exercise PDOStatement::fetch styles */
print("PDO_FETCH_ASSOC: ");
print("Return next row as an array indexed by column name\n");
$result = $sth->fetch(PDO_FETCH_ASSOC);
print_r($result);
print("\n");

print("PDO_FETCH_BOTH: ");
print("Return next row as an array indexed by both column name and number\n");
$result = $sth->fetch(PDO_FETCH_BOTH);
print_r($result);
print("\n");

print("PDO_FETCH_LAZY: ");
print("Return next row as an anonymous object with column names as properties\n");
$result = $sth->fetch(PDO_FETCH_LAZY);
print_r($result);
print("\n");

print("PDO_FETCH_OBJ: ");
print("Return next row as an anonymous object with column names as properties\n");
$result = $sth->fetch(PDO_FETCH_OBJ);
print $result->NAME;
print("\n");
?>
```

El resultado del ejemplo seria:

```
PDO_FETCH_ASSOC: Return next row as an array indexed by column name
Array
(
    [NAME] => apple
    [COLOUR] => red
)

PDO_FETCH_BOTH: Return next row as an array indexed by both column name and number
Array
(
    [NAME] => banana
    [0] => banana
    [COLOUR] => yellow
    [1] => yellow
)

PDO_FETCH_LAZY: Return next row as an anonymous object with column names as properties
PDORow Object
(
    [NAME] => orange
    [COLOUR] => orange
)

PDO_FETCH_OBJ: Return next row as an anonymous object with column names as properties
kiwi
```

Ejemplo 2. Fetching rows with a scrollable cursor

```
<?php
function readDataForwards($dbh) {
  $sql = 'SELECT hand, won, bet FROM mynumbers ORDER BY BET';
  try {
    $stmt = $dbh->prepare($sql, array(PDO_ATTR_CURSOR, PDO_CURSOR_SCROLL));
    $stmt->execute();
    while ($row = $stmt->fetch(PDO_FETCH_NUM, PDO_FETCH_ORI_NEXT)) {
      $data = $row[0] . "\t" . $row[1] . "\t" . $row[2] . "\n";
      print $data;
    }
    $stmt = null;
  }
  catch (PDOException $e) {
    print $e->getMessage();
  }
}
function readDataBackwards($dbh) {
  $sql = 'SELECT hand, won, bet FROM mynumbers ORDER BY bet';
  try {
    $stmt = $dbh->prepare($sql, array(PDO_ATTR_CURSOR, PDO_CURSOR_SCROLL));
    $stmt->execute();
    $row = $stmt->fetch(PDO_FETCH_NUM, PDO_FETCH_ORI_LAST);
    do {
      $data = $row[0] . "\t" . $row[1] . "\t" . $row[2] . "\n";
      print $data;
    } while ($row = $stmt->fetch(PDO_FETCH_NUM, PDO_FETCH_ORI_PRIOR));
    $stmt = null;
  }
  catch (PDOException $e) {
    print $e->getMessage();
  }
}

print "Reading forwards:\n";
readDataForwards($conn);

print "Reading backwards:\n";
readDataBackwards($conn);
?>
```

El resultado del ejemplo seria:

```
Reading forwards:
21    10    5
16    0     5
19    20    10

Reading backwards:
19    20    10
16    0     5
21    10    5
```

Ver también

PDO::query()
PDOStatement::fetchAll()
PDOStatement::fetchSingle()
PDOStatement::prepare()
PDOStatement::setFetchMode()

PDOStatement::fetchAll

(no version information, might be only in CVS)

PDOStatement::fetchAll -- Returns an array containing all of the result set rows

Descripción

array **PDOStatement::fetchAll** ([int fetch_style])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Lista de parámetros

fetch_style

Controls the contents of the returned array as documented in **PDOStatement::fetch()**. Defaults to *PDO_FETCH_BOTH*.

Valores retornados

PDOStatement::fetchAll() returns an array containing all of the remaining rows in the result set. The array represents each row as either an array of column values or an object with properties corresponding to each column name.

Using this method to fetch large result sets will result in a heavy demand on system and possibly network resources. Rather than retrieving all of the data and manipulating it in PHP, consider using the database server to manipulate the result sets. For example, use the WHERE and SORT BY clauses in SQL to restrict results before retrieving and processing them with PHP.

Ejemplos

Ejemplo 1. Fetch all remaining rows in a result set

```
<?php
$sth = $dbh->prepare("SELECT name, colour FROM fruit");
$sth->execute();

/* Fetch all of the remaining rows in the result set */
print("Fetch all of the remaining rows in the result set:\n");
$result = $sth->fetchAll();
print_r($result);
?>
```

El resultado del ejemplo seria:

```
Fetch all of the remaining rows in the result set:
Array
(
    [0] => Array
        (
            [NAME] => pear
            [0] => pear
            [COLOUR] => green
            [1] => green
        )

    [1] => Array
        (
            [NAME] => watermelon
            [0] => watermelon
            [COLOUR] => pink
            [1] => pink
        )

)
```

Ver también

PDO::query()
PDOStatement::fetch()
PDOStatement::fetchSingle()
PDOStatement::prepare()
PDOStatement::setFetchMode()

PDOStatement::fetchSingle

(no version information, might be only in CVS)

PDOStatement::fetchSingle -- Returns the first column in the next row of a result set

Descripción

string **PDOStatement::fetchSingle** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Valores retornados

PDOStatement::fetchSingle() returns the first column in the next row of a result set as a *string* value.

Aviso
There is no way to return the second or subsequent columns from a row if you use this method to retrieve data.

Ejemplos

Ejemplo 1. Return first column of the next row

```
<?php
$sth = $dbh->prepare("SELECT name, colour FROM fruit");
$sth->execute();

/* Fetch the first column from the next row in the result set */
print("Fetch the first column from the next row in the result set:\n");
$result = $sth->fetchSingle();
print("$result\n");

$result = $sth->fetchSingle();
print("$result\n");
?>
```

El resultado del ejemplo seria:

```
Fetch the first column from the next row in the result set:
lemon
orange
```

Ver también

PDO::query()
PDOStatement::fetch()
PDOStatement::fetchAll()
PDOStatement::prepare()
PDOStatement::setFetchMode()

PDOStatement::getAttribute

(no version information, might be only in CVS)

PDOStatement::getAttribute -- Retrieve a statement attribute

Descripción

mixed **PDOStatement::getAttribute** (long attribute)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

Ver también

PDO::getAttribute()
PDO::setAttribute()
PDOStatement::setAttribute()

PDOStatement::getColumnMeta

(no version information, might be only in CVS)

PDOStatement::getColumnMeta -- Returns meta data for a numbered column

Descripción

array **PDOStatement::getColumnMeta** (int $column)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

Lista de parámetros

$column

Its description

Valores retornados

What the function returns, first on success, then on failure. See also the Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo. entity

PDOStatement::rowCount

(no version information, might be only in CVS)

PDOStatement::rowCount -- Returns the number of rows affected by the last SQL statement

Description

int **PDOStatement::rowCount** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

PDOStatement::rowCount() returns the number of rows affected by the last DELETE, INSERT, or UPDATE statement executed by the corresponding *PDOStatement* object.

If the last SQL statement executed by the associated *PDOStatement* was a SELECT statement, some databases may return the number of rows returned by that statement. However, this behaviour is not guaranteed for all databases and should not be relied on for portable applications.

Ejemplo 1. Return the number of deleted rows

```
<?php
/* Delete all rows from the FRUIT table */
$del = $dbh->prepare('DELETE FROM fruit');
$del->execute();

/* Return number of rows that were deleted */
print("Return number of rows that were deleted:\n");
$count = $del->rowCount();
print("Deleted $count rows.\n");
?>
```

PDOStatement::setAttribute

(no version information, might be only in CVS)

PDOStatement::setAttribute -- Set a statement attribute

Descripción

bool **PDOStatement::setAttribute** (long attribute, mixed value)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

Ver también

PDO::getAttribute()
PDO::setAttribute()
PDOStatement::getAttribute()

PDOStatement::setFetchMode

(no version information, might be only in CVS)

PDOStatement::setFetchMode -- Set the default fetch mode for this statement

Descripción

bool **PDOStatement::setFetchMode** (int mode)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Lista de parámetros

mode

The fetch mode must be one of the *PDO_FETCH_** constants.

Valores retornados

Returns *1* on success or **FALSE** on failure.

Ejemplos

Ejemplo 1. Setting the fetch mode

The following example demonstrates how **PDOStatement::setFetchMode()** changes the default fetch mode for a PDOStatement object.

```
<?php
$sql = 'SELECT name, colour, calories FROM fruit';
try {
  $stmt = $dbh->query($sql);
  $result = $stmt->setFetchMode(PDO_FETCH_NUM);
  while ($row = $stmt->fetch()) {
    print $row[0] . "\t" . $row[1] . "\t" . $row[2] . "\n";
  }
}
catch (PDOException $e) {
  print $e->getMessage();
}
?>
```

El resultado del ejemplo seria:

```
apple   red     150
banana  yellow  250
orange  orange  300
kiwi    brown   75
lemon   yellow  25
pear    green   150
watermelon      pink    90
```

XCIX. Verisign Payflow Pro functions

This extension allows you to process credit cards and other financial transactions using Verisign Payment Services, formerly known as Signio (http://www.verisign.com/products/payflow/pro/index.html).

These functions are only available if PHP has been compiled with the *--with-pfpro[=DIR]* option. You will require the appropriate SDK for your platform, which may be downloaded from within the manager interface once you have registered.

Once you have downloaded the SDK you should copy the files from the `lib` directory of the distribution. Copy the header file `pfpro.h` to `/usr/local/include` and the library file `libpfpro.so` to `/usr/local/lib`.

When using these functions, you may omit calls to **pfpro_init()** and **pfpro_cleanup()** as this extension will do so automatically if required. However the functions are still available in case you are processing a number of transactions and require fine control over the library. You may perform any number of transactions using **pfpro_process()** between the two.

These functions have been added in PHP 4.0.2.

Nota: These functions only provide a link to Verisign Payment Services. Be sure to read

the Payflow Pro Developers Guide for full details of the required parameters.

Tabla de contenidos

pfpro_cleanup

(PHP 4 >= 4.0.2, PHP 5)

pfpro_cleanup -- Shuts down the Payflow Pro library

Description

void **pfpro_cleanup** (void)

pfpro_cleanup() is used to shutdown the Payflow Pro library cleanly. It should be called after you have processed any transactions and before the end of your script. However you may omit this call, in which case this extension will automatically call **pfpro_cleanup()** after your script terminates.

See also **pfpro_init()**.

pfpro_init

(PHP 4 >= 4.0.2, PHP 5)

pfpro_init -- Initialises the Payflow Pro library

Description

void **pfpro_init** (void)

pfpro_init() is used to initialise the Payflow Pro library. You may omit this call, in which case this extension will automatically call **pfpro_init()** before the first transaction.

See also **pfpro_cleanup()**.

pfpro_process_raw

(PHP 4 >= 4.0.2, PHP 5)

pfpro_process_raw -- Process a raw transaction with Payflow Pro

Description

string **pfpro_process_raw** (string parameters [, string address [, int port [, int timeout [, string proxy address [, int proxy port [, string proxy logon [, string proxy password]]]]]]])

Returns: A string containing the response.

pfpro_process_raw() processes a raw transaction string with Payflow Pro. You should really use **pfpro_process()** instead, as the encoding rules of these transactions are non-standard.

The first parameter in this case is a string containing the raw transaction request. All other parameters are the same as with **pfpro_process()**. The return value is a string containing the raw response.

> **Nota:** Be sure to read the Payflow Pro Developers Guide for full details of the required parameters and encoding rules. You would be well advised to use **pfpro_process()** instead.

Ejemplo 1. Payflow Pro raw example

```
<?php

pfpro_init();

$response = pfpro_process("USER=mylogin&PWD[5]=m&ndy&TRXTYPE=S&TENDER=C&AMT=1.50&ACCT=4

if (!$response) {
  die("Couldn't establish link to Verisign.\n");
}

echo "Verisign raw response was ".$response;

pfpro_cleanup();

?>
```

pfpro_process

(PHP 4 >= 4.0.2, PHP 5)

pfpro_process -- Process a transaction with Payflow Pro

Description

array **pfpro_process** (array parameters [, string address [, int port [, int timeout [, string proxy address [, int proxy port [, string proxy logon [, string proxy password]]]]]]])

Returns: An associative array containing the response

pfpro_process() processes a transaction with Payflow Pro. The first parameter is an associative array containing keys and values that will be encoded and passed to the processor.

The second parameter is optional and specifies the host to connect to. By default this is "test.signio.com", so you will certainly want to change this to "connect.signio.com" in order to process live transactions.

The third parameter specifies the port to connect on. It defaults to 443, the standard SSL port.

The fourth parameter specifies the timeout to be used, in seconds. This defaults to 30 seconds. Note that this timeout appears to only begin once a link to the processor has been established and so your script could potentially continue for a very long time in the event of DNS or network problems.

The fifth parameter, if required, specifies the hostname of your SSL proxy. The sixth parameter specifies the port to use.

The seventh and eighth parameters specify the logon identity and password to use on the proxy.

The function returns an associative array of the keys and values in the response.

> **Nota:** Be sure to read the Payflow Pro Developers Guide for full details of the required parameters.

Ejemplo 1. Payflow Pro example

```
<?php

pfpro_init();

$transaction = array(USER       => 'mylogin',
                     PWD        => 'mypassword',
                     TRXTYPE    => 'S',
                     TENDER     => 'C',
                     AMT        => 1.50,
                     ACCT       => '4111111111111111',
                     EXPDATE    => '0904'
                     );

$response = pfpro_process($transaction);

if (!$response) {
  die("Couldn't establish link to Verisign.\n");
}

echo "Verisign response code was ".$response[RESULT];
echo ", which means: ".$response[RESPMSG]."\n";

echo "\nThe transaction request: ";
print_r($transaction);

echo "\nThe response: ";
print_r($response);

pfpro_cleanup();

?>
```

pfpro_version

(PHP 4 >= 4.0.2, PHP 5)

pfpro_version -- Returns the version of the Payflow Pro software

Description

string **pfpro_version** (void)

pfpro_version() returns the version string of the Payflow Pro library. At the time of writing, this was L211.

C. Funciones PostgreSQL

Introducción

La base de datos PostrgeSQL es un producto Open Source y disponible sin costo. Postgres, desarrollado originalmente en el Deportamento de Ciencias de Computación de UC Berkeley, fue pionero en muchos de los conceptos de objetos y relacionales que ahora están apareciendo en algunas bases de datos comerciales. Provee soporte para lenguajes SQL92/SQL99, transacciones, integridad referencial, procedimientos almacenados y extensibilidad de tipos. PostgreSQL es un descendiente de código abierto de su código original de Berkeley.

Requirimientos

Para hacer uso del soporte PostgreSQL, necesita PostgreSQL 6.5 o posterior, PostgreSQL 7.0 o posterior para habilitar todas las características del módulo. PostgreSQL soporta varias codificaciones de caracteres, incluyendo codificación de caracteres multibyte. Su versión actual, asi como más información sobre PostgreSQL se encuentra disponible en http://www.postgresql.org/ y http://techdocs.postgresql.org/.

Instalación

In order to enable PostgreSQL support, *--with-pgsql[=DIR]* is required when you compile PHP. DIR is the PostgreSQL base install directory, defaults to `/usr/local/pgsql`. If shared object module is available, PostgreSQL module may be loaded using extension directive in `php.ini` or **dl()** function.

Configuración en tiempo de ejecución

El comportamiento de estas funciones está afectado por los valores definidos en `php.ini`.

Tabla 1. PostgreSQL configuration options

Name	Default	Changeable
pgsql.allow_persistent	"1"	PHP_INI_SYSTEM
pgsql.max_persistent	"-1"	PHP_INI_SYSTEM
pgsql.max_links	"-1"	PHP_INI_SYSTEM
pgsql.auto_reset_persistent	"0"	PHP_INI_SYSTEM
pgsql.ignore_notice	"0"	PHP_INI_ALL
pgsql.log_notice	"0"	PHP_INI_ALL

For further details and definitions of the PHP_INI_* constants, see the Apéndice H.

A continuación se presenta una corta explicación de las directivas de configuración.

pgsql.allow_persistent **boolean**

Whether to allow persistent Postgres connections.

pgsql.max_persistent **integer**

The maximum number of persistent Postgres connections per process.

pgsql.max_links **integer**

The maximum number of Postgres connections per process, including persistent connections.

pgsql.auto_reset_persistent **integer**

Detect broken persistent links with **pg_pconnect()**. Needs a little overhead.

pgsql.ignore_notice **integer**

Whether or not to ignore PostgreSQL backend notices.

pgsql.log_notice **integer**

Whether or not to log PostgreSQL backends notice messages. The PHP directive pgsql.ignore_notice must be off in order to log notice messages.

Modo de uso y consejos

Aviso
El uso del módulo PostgreSQL con PHP 4.0.6 no se recomienda debido a un fallo en el código de gestión de mensajes tipo noticia. Use la versión 4.1.0 o posterior.

Aviso
Los nombres de funciones PostgreSQL serán modificados en el lanzamiento 4.2.0 para adoptar los estándares de código actuales. La mayoría de nombres nuevos tendrán signos de subrayado adicionales, p.ej. pg_lo_open(). Algunas funciones son renombradas a nuevos nombres por razones de consistencia, p.ej. pg_exec() a pg_query(). Los nombres antiguos pueden ser usados en 4.2.0 y algunas versiones subsiguientes, pero pueden ser eliminados en el futuro. **Tabla 2. Nombres de función modificados**

Nombre antiguo	Nombre nuevo
pg_cmdtuples()	pg_affected_rows()
pg_errormessage()	pg_last_error()
pg_exec()	pg_query()
pg_fieldname()	pg_field_name()
pg_fieldsize()	pg_field_size()
pg_fieldnum()	pg_field_num()
pg_fieldprtlen()	pg_field_prtlen()
pg_fieldisnull()	pg_field_is_null()
pg_freeresult()	pg_free_result()
pg_getlastoid()	pg_last_oid()
pg_loreadall()	pg_lo_read_all()
pg_locreate()	pg_lo_create()
pg_lounlink()	pg_lo_unlink()
pg_loopen()	pg_lo_open()
pg_loclose()	pg_lo_close()
pg_loread()	pg_lo_read()
pg_lowrite()	pg_lo_write()
pg_loimport()	pg_lo_import()
pg_loexport()	pg_lo_export()
pg_numrows()	pg_num_rows()
pg_numfields()	pg_num_fields()
pg_result()	pg_fetch_result()
La vieja sintaxis pg_connect()/pg_pconnect() será marcada como obsoleta para dar soporte a conexiones asincrónicas en el futuro. Por favor use una cadena de conexión para pg_connect() y pg_pconnect().	

No todas las funciones son soportadas en todas las instalaciones. Depende de su versión de libpq (la interfaz C de Cliente PostrgeSQL) y de cómo fue compilado libpq. Si hace falta alguna función, libpq no soporta la característica requerida para la función.

También es importante que no use una versión de libpq más antigua que la del Servidor PostgreSQL al que se conectará. Si usa una versión de libpq más antigua que la que el Servidor PostgreSQL espera, puede tener problemas.

A partir de la versión 6.3 (03/02/1998) PostgreSQL usa sockets de dominio unix por defecto. El puerto TCP NO será abierto por defecto. A continuación se presenta una tabla que describe estas posibilidades de conexión nuevas. Este socket se encontrará en `/tmp/.s.PGSQL.5432`. Esta opción puede habilitarse con la bandera '-i' a **postmaster** y su significado es: "escuche en sockets TCP/IP asi como en sockets de dominio Unix".

Tabla 3. Postmaster y PHP

Postmaster	PHP	Status
postmaster &	pg_connect ("dbname=NombreDeMiBD");	OK
postmaster -i &	pg_connect ("dbname=NombreDeMiBD");	OK
postmaster &	pg_connect ("host=localhost dbname=NombreDeMiBD");	No fue posible conectarse con el servidor PostgreSQL: connectDB() falló: ¿Está corriendo postmaster y acepta conexiones TCP/IP (con -i) en 'localhost' en el puerto '5432'? en /ruta/hacia/archivo.php en la línea 20.
postmaster -i &	pg_connect ("host=localhost dbname=NombreDeMiBD");	OK

Puede establecerse una conexión con el servidor PostgreSQL con el siguiente juego de pares de valores en la cadena de comando: **$con = pg_connect("host=miHost port=miPuerto tty=miTTY options=misOpciones dbname=miBD user=miUsuario password=miContrasenya ");**

La sintaxis previa de: **$conn = pg_connect ("host", "port", "options", "tty", "dbname")** ha sido marcada como obsoleta.

Las variables de entorno afectan el comportamiente de servidor/cliente de PostgreSQL. Por ejemplo, el módulo PostgreSQL buscará la variable de entorno PGHOSN cuando el nombre de host sea omitido en la cadena de conexión. Las variables de entorno soportadas son diferentes entre versión y versión. Refiérase el Manual de Programador de PostgreSQL (libpq - Variables de Entorno) para más detalles.

Asegúrese de establecer las variables de entorno para el usuario apropiado. Use *$_ENV* o **getenv()** para chequear cuáles variables de entorno están disponibles en el proceso actual.

Ejemplo 1. Definición de parámetros predeterminados

```
PGHOST=pgsql.example.com
PGPORT=7890
PGDATABASE=sistema-web
PGUSER=usuario-web
PGPASSWORD=secreto
PGDATESTYLE=ISO
PGTZ=JST
PGCLIENTENCODING=EUC-JP

export PGHOST PGPORT PGDATABASE PGUSER PGPASSWORD PGDATESTYLE PGTZ PGCLIENTENCODING
```

> **Nota:** PostgreSQL convierte automáticamente todos los identificadores (p.ej. nombres de tablas/columnas) a valores en minúsculas. Para lograr que reconozca valores en mayúsculas, debe rodear siempre el identificador entre comillas.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

PGSQL_ASSOC (integer)

PGSQL_NUM (integer)

PGSQL_BOTH (integer)

PGSQL_CONNECTION_BAD (integer)

PGSQL_CONNECTION_OK (integer)

PGSQL_SEEK_SET (integer)

PGSQL_SEEK_CUR (integer)

PGSQL_SEEK_END (integer)

PGSQL_ESCAPE_STRING (integer)

PGSQL_ESCAPE_BYTEA (integer)

PGSQL_EMPTY_QUERY (integer)

PGSQL_COMMAND_OK (integer)

PGSQL_TUPLES_OK (integer)

PGSQL_COPY_OUT (integer)

PGSQL_COPY_IN (integer)

PGSQL_BAD_RESPONSE (integer)

PGSQL_NONFATAL_ERROR (integer)

PGSQL_FATAL_ERROR (integer)

Ejemplos

A partir de PostgreSQL 7.1.0, puede almacenar hasta 1GB en un campo de tipo texto. En versiones anteriores, éste se encontraba limitado al tamaño de bloque (por defecto 8KB, el máximo era 32KB, definido en tiempo de compilación)

Para usar la interfaz de objetos grandes (lo), es necesario ubicar las funciones de objetos grandes en el interior de un bloque de transacción. Un bloque de transacción comienza con una sentencia SQL **BEGIN** y, si la transacción fue válida, termina con **COMMIT** o **END**. Si la transacción falla, ésta debe ser cerrada con **ROLLBACK** o **ABORT**.

Ejemplo 2. Uso de Objetos Grandes

```php
<?php
    $base_de_datos = pg_connect("dbname=jacarta");
    pg_query($base_de_datos, "begin");
    $oid = pg_lo_create($base_de_datos);
    echo "$oid\n";
    $gestor = pg_lo_open($base_de_datos, $oid, "w");
    echo "$gestor\n";
    pg_lo_write($gestor, "datos de objeto grande");
    pg_lo_close($gestor);
    pg_query($base_de_datos, "commit");
?>
```

Usted no debe cerrar la conexión con el servidor PostgreSQL antes de cerrar el objeto grande.

Tabla de contenidos

pg_lo_open -- Open a large object
pg_lo_read_all -- Reads an entire large object and send straight to browser
pg_lo_read -- Read a large object
pg_lo_seek -- Seeks position of large object
pg_lo_tell -- Returns current position of large object
pg_lo_unlink -- Delete a large object
pg_lo_write -- Write a large object
pg_meta_data -- Get meta data for table
pg_num_fields -- Returns the number of fields
pg_num_rows -- Returns the number of rows
pg_Options -- Devuelve opciones
pg_parameter_status -- Returns the value of a server parameter
pg_pConnect -- Crea una conexión persistente con una base de datos
pg_ping -- Ping database connection
pg_Port -- Devuelve el número de puerto
pg_put_line -- Send a NULL-terminated string to PostgreSQL backend
pg_query -- Execute a query
pg_result_error -- Get error message associated with result
pg_result_seek -- Set internal row offset in result resource
pg_result_status -- Get status of query result
pg_select -- Select records
pg_send_query -- Sends asynchronous query
pg_set_client_encoding -- Set the client encoding
pg_trace -- Enable tracing a PostgreSQL connection
pg_tty -- Devuelve el nombre del tty
pg_unescape_bytea -- Unescape binary for bytea type
pg_untrace -- Disable tracing of a PostgreSQL connection
pg_update -- Update table
pg_version -- Returns an array with client, protocol and server version (when available)

pg_affected_rows

(PHP 4 >= 4.2.0, PHP 5)

pg_affected_rows -- Returns number of affected records (tuples)

Description

int **pg_affected_rows** (resource result)

pg_affected_rows() returns the number of tuples (instances/records/rows) affected by INSERT, UPDATE, and DELETE queries executed by **pg_query()**. If no tuple is affected by this function, it will return 0.

Ejemplo 1. pg_affected_rows() example

```
<?php
     $result = pg_query($conn, "INSERT INTO authors VALUES ('Orwell', 2002, 'Animal Far
     $cmdtuples = pg_affected_rows($result);
     echo $cmdtuples . " tuples are affected.\n";
?>
```

Nota: This function used to be called *pg_cmdtuples()*.

See also **pg_query()** and **pg_num_rows()**.

pg_cancel_query

(PHP 4 >= 4.2.0, PHP 5)

pg_cancel_query -- Cancel asynchronous query

Description

bool **pg_cancel_query** (resource connection)

pg_cancel_query() cancel asynchronous query sent by **pg_send_query()**. You cannot cancel query executed by **pg_query()**.

See also **pg_send_query()** and **pg_connection_busy()**.

pg_client_encoding

(PHP 3 CVS only, PHP 4 >= 4.0.3, PHP 5)

pg_client_encoding -- Gets the client encoding

Description

string **pg_client_encoding** ([resource connection])

pg_client_encoding() returns the client encoding as the string. The returned string should be either : SQL_ASCII, EUC_JP, EUC_CN, EUC_KR, EUC_TW, UNICODE, MULE_INTERNAL, LATINX (X=1...9), KOI8, WIN, ALT, SJIS, BIG5, WIN1250.

> **Nota:** This function requires PHP-4.0.3 or higher and PostgreSQL-7.0 or higher. If libpq is compiled without multibyte encoding support, **pg_set_client_encoding()** always return "SQL_ASCII". Supported encoding depends on PostgreSQL version. Refer to PostgreSQL manual for details to enable multibyte support and encoding supported.
>
> The function used to be called **pg_clientencoding()**.

See also **pg_set_client_encoding()**.

pg_Close

(PHP 3, PHP 4 , PHP 5)

pg_Close -- Cierra una conexión PostgreSQL

Descripción

bool **pg_close** (int connection)

Devuelve **FALSE** si connection no es un indice de conexión valido y **TRUE** en cualquier otro caso. Cierra la conexión a la base de datos PostgreSQL asociada con el indice de conexión pasado como parámetro.

pg_Connect

(PHP 3, PHP 4 , PHP 5)

pg_Connect -- Abre una conexión

Descripción

int **pg_connect** (string host, string port, string options, string tty, string dbname)

Devuelve un índice de conexión en caso de éxito, o falso si la conexión no se puede realizar. Esta función abre una conexión a una base de datos PostgreSQL. Cada uno de los argumentos debe ser una cadena entrecomillada, incluyendo el número de puerto. Los parámetros options y tty son opcionales y pueden ser omitidos. Esta función devuelve un índice de conexión que se necesitará para otras funciones PostgreSQL. Puedes tener multiples conexiones abiertas al mismo tiempo.

Una conexión también se puede establecer con el siguiente comando: **$conn = pg_connect ("dbname=marliese port=5432");** Otros parámetros aparte de *dbname* y *port* son *host*, *tty*, *options*, *user* y *password*.

Ver también **pg_pConnect()**.

pg_connection_busy

(PHP 4 >= 4.2.0, PHP 5)

pg_connection_busy -- Get connection is busy or not

Description

bool **pg_connection_busy** (resource connection)

pg_connection_busy() returns **TRUE** if the connection is busy. If it is busy, a previous query is still executing. If **pg_get_result()** is called, it will be blocked.

Ejemplo 1. pg_connection_busy() example

```
<?php
    $dbconn = pg_connect("dbname=publisher") or die("Could not connect");
    $bs = pg_connection_busy($dbconn);
    if ($bs) {
        echo 'connection is busy';
    } else {
       echo 'connection is not busy';
    }
?>
```

See also **pg_connection_status()** and **pg_get_result()**.

pg_connection_reset

(PHP 4 >= 4.2.0, PHP 5)

pg_connection_reset -- Reset connection (reconnect)

Description

bool **pg_connection_reset** (resource connection)

pg_connection_reset() resets the connection. It is useful for error recovery. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplo 1. pg_connection_reset() example

```
<?php
    $dbconn = pg_connect("dbname=publisher") or die("Could not connect");
    $dbconn2 = pg_connection_reset($dbconn);
    if ($dbconn2) {
        echo "reset successful\n";
    } else {
        echo "reset failed\n";
    }
?>
```

See also **pg_connect()**, **pg_pconnect()** and **pg_connection_status()**.

pg_connection_status

(PHP 4 >= 4.2.0, PHP 5)

pg_connection_status -- Get connection status

Description

int **pg_connection_status** (resource connection)

pg_connection_status() returns a connection status. Possible statuses are *PGSQL_CONNECTION_OK* and *PGSQL_CONNECTION_BAD*. The return value 0 as integer indicates a valid connection.

Ejemplo 1. pg_connection_status() example

```
<?php
    $dbconn = pg_connect("dbname=publisher") or die("Could not connect");
    $stat = pg_connection_status($dbconn);
    if ($stat === 0) {
        echo 'Connection status ok';
    } else {
        echo 'Connection status bad';
    }
?>
```

See also **pg_connection_busy()**.

pg_convert

(PHP 4 >= 4.3.0, PHP 5)

pg_convert -- Convert associative array value into suitable for SQL statement

Description

array **pg_convert** (resource connection, string table_name, array assoc_array [, int options])

pg_convert() checks and converts the values in *assoc_array* into suitable values for use in a SQL statement. Precondition for **pg_convert()** is the existence of a table *table_name* which has at least as many columns as *assoc_array* has elements. The fieldnames as well as the fieldvalues in *table_name* must match the indices and values of *assoc_array*. Returns an array with the converted values on success, **FALSE** otherwise.

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

See also **pg_meta_data()**.

pg_copy_from

(PHP 4 >= 4.2.0, PHP 5)

pg_copy_from -- Insert records into a table from an array

Description

bool **pg_copy_from** (resource connection, string table_name, array rows [, string delimiter [, string null_as]])

pg_copy_from() insert records into a table from *rows*. It issues *COPY FROM* SQL command internally to insert records. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

See also **pg_copy_to()**.

pg_copy_to

(PHP 4 >= 4.2.0, PHP 5)

pg_copy_to -- Copy a table to an array

Description

array **pg_copy_to** (resource connection, string table_name [, string delimiter [, string null_as]])

pg_copy_to() copies a table to an array. It issues *COPY TO* SQL command internally to retrieve records. The resulting array is returned. It returns **FALSE** on failure.

See also **pg_copy_from()**.

pg_DBname

(PHP 3, PHP 4 , PHP 5)

pg_DBname -- Nombre de la base de datos

Descripción

string **pg_dbname** (int connection)

Devuelve el nombre de la base de datos a la cual es el índice de conexión con PostgreSQL está conectado, o **FALSE** si connection no es un índice de conexión válido.

pg_delete

(PHP 4 >= 4.3.0, PHP 5)

pg_delete -- Deletes records

Description

mixed **pg_delete** (resource connection, string table_name, array assoc_array [, int options])

pg_delete() deletes record condition specified by *assoc_array* which has *field=>value*. If *option* is specified, **pg_convert()** is applied to *assoc_array* with specified option.

Ejemplo 1. pg_delete() example

```
<?php
    $db = pg_connect('dbname=foo');
    // This is safe, since $_POST is converted automatically
    $res = pg_delete($db, 'post_log', $_POST);
    if ($res) {
        echo "POST data is deleted: $res\n";
    } else {
        echo "User must have sent wrong inputs\n";
    }
?>
```

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

See also **pg_convert()**.

pg_end_copy

(PHP 4 >= 4.0.3, PHP 5)

pg_end_copy -- Sync with PostgreSQL backend

Description

bool **pg_end_copy** ([resource connection])

pg_end_copy() syncs the PostgreSQL frontend (usually a web server process) with the PostgreSQL server after doing a copy operation performed by **pg_put_line()**. **pg_end_copy()** must be issued, otherwise the PostgreSQL server may get out of sync with the frontend and will report an error. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

For further details and an example, see also **pg_put_line()**.

pg_escape_bytea

(PHP 4 >= 4.2.0, PHP 5)

pg_escape_bytea -- Escape binary for bytea type

Description

string **pg_escape_bytea** (string data)

pg_escape_bytea() escapes string for bytea datatype. It returns escaped string.

> **Nota:** When you SELECT bytea type, PostgreSQL returns octal byte value prefixed by \ (e.g. \032). Users are supposed to convert back to binary format by yourself.
>
> This function requires PostgreSQL 7.2 or later. With PostgreSQL 7.2.0 and 7.2.1, bytea type must be casted when you enable multi-byte support. i.e. *INSERT INTO test_table (image) VALUES ('$image_escaped'::bytea);* PostgreSQL 7.2.2 or later does not need cast. Exception is when client and backend character encoding does not match, there may be multi-byte stream error. User must cast to bytea to avoid this error.

See also **pg_unescape_bytea()** and **pg_escape_string()**.

pg_escape_string

(PHP 4 >= 4.2.0, PHP 5)

pg_escape_string -- Escape string for text/char type

Description

string **pg_escape_string** (string data)

pg_escape_string() escapes string for text/char datatype. It returns escaped string for PostgreSQL. Use of this function is recommended instead of **addslashes()**.

> **Nota:** This function requires PostgreSQL 7.2 or later.

See also **pg_escape_bytea()**

pg_fetch_all

(PHP 4 >= 4.3.0, PHP 5)

pg_fetch_all -- Fetches all rows from a result as an array

Description

array **pg_fetch_all** (resource result)

pg_fetch_all() returns an array that contains all rows (tuples/records) in result resource. It returns **FALSE**, if there are no rows.

> **Nota:** Esta funcion define campos NULL como valores PHP **NULL**.

Ejemplo 1. PostgreSQL fetch all

```
<?php
$conn = pg_pconnect("dbname=publisher");
if (!$conn) {
    echo "An error occured.\n";
    exit;
}

$result = pg_query($conn, "SELECT * FROM authors");
if (!$result) {
    echo "An error occured.\n";
    exit;
}

$arr = pg_fetch_all($result);

var_dump($arr);

?>
```

See also **pg_fetch_row()**, **pg_fetch_array()**, **pg_fetch_object()** and **pg_fetch_result()**.

pg_Fetch_Array

(PHP 3>= 3.0.1, PHP 4 , PHP 5)

pg_Fetch_Array -- obtiene una fila en la forma de un array

Descripción

array **pg_fetch_array** (int result, int row [, int result_type])

Devuelve: Un array que se corresponde con la fila obtenida, o **FALSE** si no hay más filas.

pg_fetch_array() es una versión extendida de **pg_fetch_row()**. Además de almacenar los datos en los índices numericos del array resultante, también almacena los datos usando índices asociativos, empleando para ello el nombre del campo como la llave o índice.

El tercer parámetro opcional *result_type* en **pg_fetch_array()** es una constante y puede tomar cualquiera de los siguientes valores: PGSQL_ASSOC, PGSQL_NUM, y PGSQL_BOTH.

> **Nota:** *Result_type* se añadio en PHP 4.0.

Una cosa importante a tener en cuenta es que usar **pg_fetch_array()** NO es significativamente más lento que usar **pg_fetch_row()**, y sin embargo el valor añadido que aporta sí lo es.

Para más detalles, ver **pg_fetch_row()**

Ejemplo 1. PostgreSQL fetch array

```
<?php
$conn = pg_pconnect("","","","","publisher");
if (!$conn) {
    echo "An error occured.\n";
    exit;
}

$result = pg_Exec ($conn, "SELECT * FROM authors");
if (!$result) {
    echo "An error occured.\n";
    exit;
}

$arr = pg_fetch_array ($result, 0);
echo $arr[0] . " <- array\n";

$arr = pg_fetch_array ($result, 1);
echo $arr["author"] . " <- array\n";
?>
```

pg_fetch_assoc

(PHP 4 >= 4.3.0, PHP 5)

pg_fetch_assoc -- Fetch a row as an associative array

Description

array **pg_fetch_assoc** (resource result [, int row])

pg_fetch_assoc() returns an associative array that corresponds to the fetched row (tuples/records). It returns **FALSE**, if there are no more rows.

pg_fetch_assoc() is ñalent to calling **pg_fetch_array()** with PGSQL_ASSOC for the optional third parameter. It only returns an associative array. If you need the numeric indices, use **pg_fetch_row()**.

> **Nota:** Esta funcion define campos NULL como valores PHP **NULL**.

row is row (record) number to be retrieved. First row is 0.

pg_fetch_assoc() is NOT significantly slower than using **pg_fetch_row()**, while it provides a

significant ease of use.

Ejemplo 1. pg_fetch_assoc() example

```
<?php
$conn = pg_connect("dbname=publisher");
if (!$conn) {
    echo "An error occured.\n";
    exit;
}

$result = pg_query($conn, "SELECT id, author, email FROM authors");
if (!$result) {
    echo "An error occured.\n";
    exit;
}

while ($row = pg_fetch_assoc($result)) {
    echo $row['id'];
    echo $row['author'];
    echo $row['email'];
}
?>
```

> **Nota:** From 4.1.0, *row* became optional. Calling **pg_fetch_assoc()** will increment the internal row counter by one.

See also **pg_fetch_row()**, **pg_fetch_array()**, **pg_fetch_object()** and **pg_fetch_result()**.

pg_Fetch_Object

(PHP 3>= 3.0.1, PHP 4 , PHP 5)

pg_Fetch_Object -- obtener una fila en forma de objeto

Descripción

object **pg_fetch_object** (int result, int row [, int result_type])

Devuelve: Un objeto cuyas propiedades se corresponden con los campos de la fila obtenida, o **FALSE** si no hay más filas.

pg_fetch_object() es parecida a **pg_fetch_array()**, con una diferencia - se devuelve un objeto, en vez de un array. Indirectamente, eso significa que solo puedes acceder a los datos por medio de su nombre de campo, y no a través de sus posiciones (los números son nombres de propiedad invalidos).

El tercer parámetro opcional *result_type* en **pg_fetch_object()** es una constante y puede tomar cualquiera de los siguientes valores: PGSQL_ASSOC, PGSQL_NUM, y PGSQL_BOTH.

> **Nota:** *Result_type* se añadio en PHP 4.0.

Referente a la velocidad, la función es identica a **pg_fetch_array()**, y practicamente tan rápida como **pg_fetch_row()** (la diferencia es insignificante).

Ver también: **pg_fetch_array()** y **pg_fetch_row()**.

Ejemplo 1. Postgres fetch object

```
<?php
$database = "verlag";
$db_conn = pg_connect ("localhost", "5432", "", "", $database);
if (!$db_conn): ?>
    <H1>Failed connecting to postgres database <? echo $database ?></H1> <?
    exit;
endif;

$qu = pg_exec ($db_conn, "SELECT * FROM verlag ORDER BY autor");
$row = 0; // postgres needs a row counter other dbs might not

while ($data = pg_fetch_object ($qu, $row)):
    echo $data->autor." (";
    echo $data->jahr ."): ";
    echo $data->titel."<BR>";
    $row++;
endwhile; ?>

<PRE><?php
$fields[] = Array ("autor", "Author");
$fields[] = Array ("jahr",  "  Year");
$fields[] = Array ("titel", " Title");

$row= 0; // postgres needs a row counter other dbs might not
while ($data = pg_fetch_object ($qu, $row)):
    echo "----------\n";
    reset ($fields);
    while (list (,$item) = each ($fields)):
        echo $item[1].": ".$data->$item[0]."\n";
    endwhile;
    $row++;
endwhile;
echo "----------\n"; ?>
</PRE> <?php
pg_freeResult ($qu);
pg_close ($db_conn);
?>
```

pg_fetch_result

(PHP 4 >= 4.2.0, PHP 5)

pg_fetch_result -- Returns values from a result resource

Description

mixed **pg_fetch_result** (resource result, int row, mixed field)

mixed **pg_fetch_result** (resource result, mixed field)

pg_fetch_result() returns values from a *result* resource returned by **pg_query()**. *row* is integer. *field* is field name (string) or field index (integer). The *row* and *field* specify what cell in the table of results to return. Row numbering starts from 0. Instead of naming the field, you may use the field index as an unquoted number. Field indices start from 0.

PostgreSQL has many built in types and only the basic ones are directly supported here. All forms of **integer** types are returned as **integer** values. All forms of float, and real types are returned as **float** values. Boolean is returned as "t" or "f". All other types, including arrays are returned as strings formatted in the same default PostgreSQL manner that you would see in the **psql** program.

pg_Fetch_Row

(PHP 3>= 3.0.1, PHP 4 , PHP 5)

pg_Fetch_Row -- obtiene la fila como un array enumerado

Descripción

array **pg_fetch_row** (int result, int row)

Devuelve: Un array que se corresponde con la fila obtenida, o **FALSE** en el caso de que no haya más filas.

pg_fetch_row() obtiene una fila de datos a partir del resultado asociado con el identificador de resultado especificado. La fila se devuelve en forma de array. Cada columna del resultado se almacena en una posición del array, empezando a partir de la posición 0.

Las siguientes llamadas a **pg_fetch_row()** devolverán la fila siguiente en el conjunto resultado, o falso en el caso de que no haya más filas que devolver.

Ver también: pg_fetch_array(), pg_fetch_object(), **pg_result()**.

Ejemplo 1. Postgres fetch row

```
<?php
$conn = pg_pconnect("","","","","publisher");
if (!$conn) {
    echo "An error occured.\n";
    exit;
}

$result = pg_Exec ($conn, "SELECT * FROM authors");
if (!$result) {
    echo "An error occured.\n";
    exit;
}

$row = pg_fetch_row ($result, 0);
echo $row[0] . " <- row\n";

$row = pg_fetch_row ($result, 1);
echo $row[0] . " <- row\n";

$row = pg_fetch_row ($result, 2);
echo $row[1] . " <- row\n";
?>
```

pg_field_is_null

(PHP 4 >= 4.2.0, PHP 5)

pg_field_is_null -- Test if a field is **NULL**

Description

int **pg_field_is_null** (resource result, int row, mixed field)

pg_field_is_null() tests if a field is **NULL** or not. It returns 1 if the field in the given row is **NULL**. It

returns 0 if the field in the given row is NOT **NULL**. Field can be specified as column index (number) or fieldname (string). Row numbering starts at 0.

Ejemplo 1. pg_field_is_null() example

```
<?php
    $dbconn = pg_connect("dbname=publisher") or die ("Could not connect");
    $res = pg_query($dbconn, "select * from authors where author = 'Orwell'");
    if ($res) {
        if (pg_field_is_null($res, 0, "year") == 1) {
            echo "The value of the field year is null.\n";
        }
        if (pg_field_is_null($res, 0, "year") == 0) {
            echo "The value of the field year is not null.\n";
      }
   }
?>
```

Nota: This function used to be called *pg_fieldisnull()*.

pg_field_name

(PHP 4 >= 4.2.0, PHP 5)

pg_field_name -- Returns the name of a field

Description

string **pg_field_name** (resource result, int field_number)

pg_field_name() returns the name of the field occupying the given *field_number* in the given PostgreSQL *result* resource. Field numbering starts from 0.

Ejemplo 1. Getting information about fields

```
<?php
    $dbconn = pg_connect("dbname=publisher") or die("Could not connect");

    $res = pg_query($dbconn, "select * from authors where author = 'Orwell'");
    $i = pg_num_fields($res);
    for ($j = 0; $j < $i; $j++) {
        echo "column $j\n";
        $fieldname = pg_field_name($res, $j);
        echo "fieldname: $fieldname\n";
        echo "printed length: " . pg_field_prtlen($res, $fieldname) . " characters\n";
        echo "storage length: " . pg_field_size($res, $j) . " bytes\n";
        echo "field type: " . pg_field_type($res, $j) . " \n\n";
    }
?>
```

The above example would produce the following output:

```
column 0
fieldname: author
printed length: 6 characters
storage length: -1 bytes
field type: varchar

column 1
fieldname: year
printed length: 4 characters
storage length: 2 bytes
field type: int2

column 2
fieldname: title
printed length: 24 characters
storage length: -1 bytes
field type: varchar
```

Nota: This function used to be called *pg_fieldname()*.

See also **pg_field_num()**.

pg_field_num

(PHP 4 >= 4.2.0, PHP 5)

pg_field_num -- Returns the field number of the named field

Description

int **pg_field_num** (resource result, string field_name)

pg_field_num() will return the number of the column (field) slot that corresponds to the *field_name* in the given PostgreSQL *result* resource. Field numbering starts at 0. This function will return -1 on error.

See the example given at the **pg_field_name()** page.

Nota: This function used to be called *pg_fieldnum()*.

See also **pg_field_name()**.

pg_field_prtlen

(PHP 4 >= 4.2.0, PHP 5)

pg_field_prtlen -- Returns the printed length

Description

int **pg_field_prtlen** (resource result [, int row_number, mixed field_name_or_number])

pg_field_prtlen() returns the actual printed length (number of characters) of a specific value in a PostgreSQL *result*. Row numbering starts at 0. This function will return -1 on an error.

field_name_or_number can be passed either as an **integer** or as a **string**. If it is passed as an **integer**, PHP recognises it as the field number, otherwise as field name.

See the example given at the **pg_field_name()** page.

> **Nota:** This function used to be called *pg_fieldprtlen()*.

See also **pg_field_size()**.

pg_field_size

(PHP 4 >= 4.2.0, PHP 5)

pg_field_size -- Returns the internal storage size of the named field

Description

int **pg_field_size** (resource result, int field_number)

pg_field_size() returns the internal storage size (in bytes) of the field number in the given PostgreSQL *result*. Field numbering starts at 0. A field size of -1 indicates a variable length field. This function will return **`FALSE`** on error.

See the example given at the **pg_field_name()** page.

> **Nota:** This function used to be called *pg_fieldsize()*.

See also **pg_field_prtlen()** and **pg_field_type()**.

pg_field_type_oid

(no version information, might be only in CVS)

pg_field_type_oid -- Returns the type ID (OID) for the corresponding field number

Description

int **pg_field_type_oid** (resource result, int field_number)

pg_field_type_oid() returns an integer containing the type ID the given *field_number* in the given PostgreSQL *result* resource. Field numbering starts at 0.

You can get more information about the field type by querying PostgreSQL pg_type system table using the ID obtained with this function.

See also **pg_field_type()**, **pg_field_prtlen()** and **pg_field_name()**.

pg_field_type

(PHP 4 >= 4.2.0, PHP 5)

pg_field_type -- Returns the type name for the corresponding field number

Description

string **pg_field_type** (resource result, int field_number)

pg_field_type() returns a string containing the type name of the given *field_number* in the given PostgreSQL *result* resource. Field numbering starts at 0.

See the example given at the **pg_field_name()** page.

> **Nota:** This function used to be called *pg_fieldtype()*.

See also **pg_field_prtlen()**, **pg_field_name()** and **pg_field_type_oid()**.

pg_free_result

(PHP 4 >= 4.2.0, PHP 5)

pg_free_result -- Free result memory

Description

bool **pg_free_result** (resource result)

pg_free_result() only needs to be called if you are worried about using too much memory while your script is running. All result memory will automatically be freed when the script is finished. But, if you are sure you are not going to need the result data anymore in a script, you may call **pg_free_result()** with the *result* resource as an argument and the associated result memory will be freed. Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

> **Nota:** This function used to be called *pg_freeresult()*.

See also **pg_query()**.

pg_get_notify

(PHP 4 >= 4.3.0, PHP 5)

pg_get_notify -- Ping database connection

Description

array **pg_get_notify** (resource connection [, int result_type])

pg_get_notify() gets notify message sent by *NOTIFY* SQL command. To receive notify messages, *LISTEN* SQL command must be issued. If there is notify message on the connection, array contains message name and backend PID is returned. If there is no message, **`FALSE`** is returned.

See also **pg_get_pid()**

Ejemplo 1. PostgreSQL NOTIFY message

```
<?php
$conn = pg_pconnect("dbname=publisher");
if (!$conn) {
    echo "An error occured.\n";
    exit;
}

// Listen 'author_updated' message from other processes
pg_query($conn, 'LISTEN author_updated;');
$notify = pg_get_notify($conn);
if (!$notify) {
    echo "No messages\n";
} else {
    print_r($notify);
}
?>
```

pg_get_pid

(PHP 4 >= 4.3.0, PHP 5)

pg_get_pid -- Ping database connection

Description

int **pg_get_pid** (resource connection)

pg_get_pid() gets backend (database server process) PID. PID is useful to check if *NOTIFY* message is sent from other process or not.

Ejemplo 1. PostgreSQL backend PID

```
<?php
$conn = pg_pconnect("dbname=publisher");
if (!$conn) {
    echo "An error occured.\n";
    exit;
}

// Backend process PID. Use PID with pg_get_notify()
$pid = pg_get_pid($conn);
?>
```

See also **pg_get_notify()**.

pg_get_result

(PHP 4 >= 4.2.0, PHP 5)

pg_get_result -- Get asynchronous query result

Description

resource **pg_get_result** ([resource connection])

pg_get_result() get result resource from async query executed by **pg_send_query()**.
pg_send_query() can send multiple queries to PostgreSQL server and **pg_get_result()** is used to

get query result one by one. It returns result resource. If there is no more results, it returns **FALSE**.

pg_Host

(PHP 3, PHP 4 , PHP 5)

pg_Host -- Devuelve el nombre del host

Descripción

string **pg_host** (int connection_id)

pg_Host() devuelve el nombre del host al que identificador conexión PostgreSQL pasado está conectado.

pg_insert

(PHP 4 >= 4.3.0, PHP 5)

pg_insert -- Insert array into table

Description

bool **pg_insert** (resource connection, string table_name, array assoc_array [, int options])

pg_insert() inserts the values of *assoc_array* into the table specified by *table_name*. *table_name* must at least have as many columns as *assoc_array* has elements. The fieldnames as well as the fieldvalues in *table_name* must match the indices and values of *assoc_array*. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo. If *options* is specified, **pg_insert()** is applied to *assoc_array* with specified option.

Ejemplo 1. pg_insert() example

```
<?php
    $dbconn = pg_connect('dbname=foo');
    // This is safe, since $_POST is converted automatically
    $res = pg_insert($dbconn, 'post_log', $_POST);
    if ($res) {
        echo "POST data is successfully logged\n";
    } else {
        echo "User must have sent wrong inputs\n";
    }
?>
```

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

See also **pg_convert()**.

pg_last_error

(PHP 4 >= 4.2.0, PHP 5)

pg_last_error -- Get the last error message string of a connection

Description

string **pg_last_error** ([resource connection])

pg_last_error() returns the last error message for given *connection*.

Error messages may be overwritten by internal PostgreSQL(libpq) function calls. It may not return appropriate error message, if multiple errors are occurred inside a PostgreSQL module function.

Use **pg_result_error()**, **pg_result_status()** and **pg_connection_status()** for better error handling.

> **Nota:** This function used to be called *pg_errormessage()*.

See also **pg_result_error()**.

pg_last_notice

(PHP 4 >= 4.0.6, PHP 5)

pg_last_notice -- Returns the last notice message from PostgreSQL server

Description

string **pg_last_notice** (resource connection)

pg_last_notice() returns the last notice message from the PostgreSQL server specified by *connection*. The PostgreSQL server sends notice messages in several cases, e.g. if the transactions can't be continued. With **pg_last_notice()**, you can avoid issuing useless queries, by checking whether the notice is related to the transaction or not.

Aviso
This function is EXPERIMENTAL and it is not fully implemented yet. **pg_last_notice()** was added in PHP 4.0.6. However, PHP 4.0.6 has problem with notice message handling. Use of the PostgreSQL module with PHP 4.0.6 is not recommended even if you are not using **pg_last_notice()**. This function is fully implemented in PHP 4.3.0. PHP earlier than PHP 4.3.0 ignores database connection parameter.

Notice message tracking can be set to optional by setting 1 for *pgsql.ignore_notice* in `php.ini` from PHP 4.3.0.

Notice message logging can be set to optional by setting 0 for *pgsql.log_notice* in `php.ini` from PHP 4.3.0. Unless *pgsql.ignore_notice* is set to 0, notice message cannot be logged.

See also **pg_query()** and **pg_last_error()**.

pg_last_oid

(PHP 4 >= 4.2.0, PHP 5)

pg_last_oid -- Returns the last object's oid

Description

int **pg_last_oid** (resource result)

pg_last_oid() is used to retrieve the *oid* assigned to an inserted tuple (record) if the result resource is used from the last command sent via **pg_query()** and was an SQL INSERT. Returns a positive integer if there was a valid *oid*. It returns **FALSE** if an error occurs or the last command sent via **pg_query()** was not an INSERT or INSERT is failed.

OID field became an optional field from PostgreSQL 7.2. When OID field is not defined in a table, programmer must use **pg_result_status()** to check if record is is inserted successfully or not.

> **Nota:** This function used to be called *pg_getlastoid()*.

See also **pg_query()** and **pg_result_status()**

pg_lo_close

(PHP 4 >= 4.2.0, PHP 5)

pg_lo_close -- Close a large object

Description

bool **pg_lo_close** (resource large_object)

pg_lo_close() closes a Large Object. *large_object* is a resource for the large object from **pg_lo_open()**.

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

> **Nota:** This function used to be called *pg_loclose()*.

See also **pg_lo_open()**, **pg_lo_create()** and **pg_lo_import()**.

pg_lo_create

(PHP 4 >= 4.2.0, PHP 5)

pg_lo_create -- Create a large object

Description

int **pg_lo_create** ([resource connection])

pg_lo_create() creates a Large Object and returns the *oid* of the large object. *connection* specifies a valid database connection opened by **pg_connect()** or **pg_pconnect()**. PostgreSQL access modes INV_READ, INV_WRITE, and INV_ARCHIVE are not supported, the object is created always with both read and write access. INV_ARCHIVE has been removed from PostgreSQL itself (version 6.3 and above). It returns large object oid, otherwise it returns **FALSE** if an error occurred.

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

> **Nota:** This function used to be called *pg_locreate()*.

pg_lo_export

(PHP 4 >= 4.2.0, PHP 5)

pg_lo_export -- Export a large object to file

Description

bool **pg_lo_export** ([resource connection, int oid, string pathname])

The *oid* argument specifies oid of the large object to export and the *pathname* argument specifies the pathname of the file. It returns **FALSE** if an error occurred, **TRUE** otherwise.

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

> **Nota:** This function used to be called *pg_loexport()*.

See also **pg_lo_import()**.

pg_lo_import

(PHP 4 >= 4.2.0, PHP 5)

pg_lo_import -- Import a large object from file

Description

int **pg_lo_import** ([resource connection, string pathname])

In versions before PHP 4.2.0 the syntax of this function was different, see the following definition:

int **pg_lo_import** (string pathname [, resource connection])

The *pathname* argument specifies the pathname of the file to be imported as a large object. It returns **FALSE** if an error occurred, oid of the just created large object otherwise.

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

> **Nota:** Cuando safe-mode (modo-seguro) está activado, PHP comprueba si los archivos o directorios que va a utilizar tienen la misma UID que el script que está siendo ejecutado.

> **Nota:** This function used to be called *pg_loimport()*.

See also **pg_lo_export()** and **pg_lo_open()**.

pg_lo_open

(PHP 4 >= 4.2.0, PHP 5)

pg_lo_open -- Open a large object

Description

resource **pg_lo_open** (resource connection, int oid, string mode)

pg_lo_open() opens a Large Object and returns large object resource. The resource encapsulates information about the connection. *oid* specifies a valid large object oid and *mode* can be either "r", "w", or "rw". It returns **FALSE** if there is an error.

Aviso
Do not close the database connection before closing the large object resource.

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

> **Nota:** This function used to be called *pg_loopen()*.

See also **pg_lo_close()** and **pg_lo_create()**.

pg_lo_read_all

(PHP 4 >= 4.2.0, PHP 5)

pg_lo_read_all -- Reads an entire large object and send straight to browser

Description

int **pg_lo_read_all** (resource large_object)

pg_lo_read_all() reads a large object and passes it straight through to the browser after sending all pending headers. Mainly intended for sending binary data like images or sound. It returns number of bytes read. It returns **FALSE**, if an error occurred.

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

> **Nota:** This function used to be called *pg_loreadall()*.

See also **pg_lo_read()**.

pg_lo_read

(PHP 4 >= 4.2.0, PHP 5)

pg_lo_read -- Read a large object

Description

string **pg_lo_read** (resource large_object [, int len])

pg_lo_read() reads at most *len* (defaults to 8192) bytes from a large object and returns it as a string. *large_object* specifies a valid large object resource and*len* specifies the maximum allowable size of the large object segment. It returns **FALSE** if there is an error.

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

> **Nota:** This function used to be called *pg_loread()*.

See also **pg_lo_read_all()**.

pg_lo_seek

(PHP 4 >= 4.2.0, PHP 5)

pg_lo_seek -- Seeks position of large object

Description

bool **pg_lo_seek** (resource large_object, int offset [, int whence])

pg_lo_seek() seeks position of large object resource. *whence* is PGSQL_SEEK_SET, PGSQL_SEEK_CUR or PGSQL_SEEK_END.

See also **pg_lo_tell()**.

pg_lo_tell

(PHP 4 >= 4.2.0, PHP 5)

pg_lo_tell -- Returns current position of large object

Description

int **pg_lo_tell** (resource large_object)

pg_lo_tell() returns current position (offset from the beginning of large object).

See also **pg_lo_seek()**.

pg_lo_unlink

(PHP 4 >= 4.2.0, PHP 5)

pg_lo_unlink -- Delete a large object

Description

bool **pg_lo_unlink** (resource connection, int oid)

pg_lo_unlink() deletes a large object with the *oid*. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

> **Nota:** This function used to be called *pg_lo_unlink()*.

See also **pg_lo_create()** and **pg_lo_import()**.

pg_lo_write

(PHP 4 >= 4.2.0, PHP 5)

pg_lo_write -- Write a large object

Description

int **pg_lo_write** (resource large_object, string data [, int len])

pg_lo_write() writes at most to a large object from a variable *data* and returns the number of bytes actually written, or **FALSE** in the case of an error. *large_object* is a large object resource from **pg_lo_open()**.

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

> **Nota:** This function used to be called *pg_lowrite()*.

See also **pg_lo_create()** and **pg_lo_open()**.

pg_meta_data

(PHP 4 >= 4.3.0, PHP 5)

pg_meta_data -- Get meta data for table

Description

array **pg_meta_data** (resource connection, string table_name)

pg_meta_data() returns table definition for *table_name* as an array. If there is error, it returns **FALSE**

Ejemplo 1. Getting table metadata

```
<?php
    $dbconn = pg_connect("dbname=publisher") or die("Could not connect");

    $meta = pg_meta_data($dbconn, 'authors');
    if (is_array($meta)) {
        echo '<pre>';
        var_dump($meta);
        echo '</pre>';
    }
?>
```

The above example would produce the following output:

```
array(3) {
  ["author"]=>
  array(5) {
    ["num"]=>
    int(1)
    ["type"]=>
    string(7) "varchar"
    ["len"]=>
    int(-1)
    ["not null"]=>
    bool(false)
    ["has default"]=>
    bool(false)
  }
  ["year"]=>
  array(5) {
    ["num"]=>
    int(2)
    ["type"]=>
    string(4) "int2"
    ["len"]=>
    int(2)
    ["not null"]=>
    bool(false)
    ["has default"]=>
    bool(false)
  }
  ["title"]=>
  array(5) {
    ["num"]=>
    int(3)
    ["type"]=>
    string(7) "varchar"
    ["len"]=>
    int(-1)
    ["not null"]=>
    bool(false)
    ["has default"]=>
    bool(false)
  }
}
```

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

See also **pg_convert()**.

pg_num_fields

(PHP 4 >= 4.2.0, PHP 5)

pg_num_fields -- Returns the number of fields

Description

int **pg_num_fields** (resource result)

pg_num_fields() returns the number of fields (columns) in a PostgreSQL *result*. The argument is a result resource returned by **pg_query()**. This function will return -1 on error.

> **Nota:** This function used to be called *pg_numfields()*.

See also **pg_num_rows()** and **pg_affected_rows()**.

pg_num_rows

(PHP 4 >= 4.2.0, PHP 5)

pg_num_rows -- Returns the number of rows

Description

int **pg_num_rows** (resource result)

pg_num_rows() will return the number of rows in a PostgreSQL *result* resource. *result* is a query result resource returned by **pg_query()**. This function will return -1 on error.

> **Nota:** Use **pg_affected_rows()** to get number of rows affected by INSERT, UPDATE and DELETE query.

> **Nota:** This function used to be called *pg_numrows()*.

See also **pg_num_fields()** and **pg_affected_rows()**.

pg_Options

(PHP 3, PHP 4 , PHP 5)

pg_Options -- Devuelve opciones

Descripción

string **pg_options** (int connection_id)

pg_Options() devuelve una cadena que contiene las opciones especificadas en el identificador de conexión con PostgreSQL dado.

pg_parameter_status

(PHP 5)

pg_parameter_status -- Returns the value of a server parameter

Description

string **pg_parameter_status** ([resource connection, string param_name])

pg_parameter_status() returns a string with the current *param_name* value. Returns **FALSE** on failure.

The parameters currently available include: server_version, client_encoding, is_superuser, session_authorization, and DateStyle.

pg_pConnect

(PHP 3, PHP 4 , PHP 5)

pg_pConnect -- Crea una conexión persistente con una base de datos

Descripción

int **pg_pconnect** (string host, string port, string options, string tty, string dbname)

Devuelve un índice de conexión en caso de éxito, o **FALSE** si no es posible realizar la conexión. Abre una conexión persistente hacia una base de datos de PostgreSQL. Cada uno de los parámetros puede ser una cadena entrecomillada (quoted), incluyendo el número de puerto. Los parámetros options y tty son opcionales y pueden omitirse. Esta función devuelve un índice de conexión que luego será empleado al llamar a otras funciones PostgreSQL. Puedes tener multiples conexiones persistentes abiertas al mismo tiempo. Ver también **pg_Connect()**.

Una conexión también se puede establecer con el comando siguiente: **$conn = pg_pconnect ("dbname=marliese port=5432");** Otros parámetros además de *dbname* y *port* son *host*, *tty*, *options*, *user* y *password*.

pg_ping

(PHP 4 >= 4.3.0, PHP 5)

pg_ping -- Ping database connection

Description

bool **pg_ping** (resource connection)

pg_ping() ping database connection, try to reconnect if it is broken. It returns **TRUE** if connection is alive, otherwise **FALSE**.

Ejemplo 1. pg_ping() example

```
<?php
$conn = pg_pconnect("dbname=publisher");
if (!$conn) {
    echo "An error occured.\n";
    exit;
}

if (!pg_ping($conn))
    die("Connection is broken\n");
?>
```

See also pg_connection_status() and pg_connection_reset().

pg_Port

(PHP 3, PHP 4 , PHP 5)

pg_Port -- Devuelve el número de puerto

Descripción

int **pg_port** (int connection_id)

pg_Port() devuelve el número del puerto al que el identificador de conexión con PostgreSQL está conectado.

pg_put_line

(PHP 4 >= 4.0.3, PHP 5)

pg_put_line -- Send a NULL-terminated string to PostgreSQL backend

Description

bool **pg_put_line** (string data)

bool **pg_put_line** (resource connection, string data)

pg_put_line() sends a NULL-terminated string to the PostgreSQL backend server. This is useful for example for very high-speed inserting of data into a table, initiated by starting a PostgreSQL copy-operation. That final NULL-character is added automatically. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

> **Nota:** The application must explicitly send the two characters "\." on the last line to indicate to the backend that it has finished sending its data.

Ejemplo 1. High-speed insertion of data into a table

```
<?php
    $conn = pg_pconnect("dbname=foo");
    pg_query($conn, "create table bar (a int4, b char(16), d float8)");
    pg_query($conn, "copy bar from stdin");
    pg_put_line($conn, "3\thello world\t4.5\n");
    pg_put_line($conn, "4\tgoodbye world\t7.11\n");
    pg_put_line($conn, "\\.\n");
    pg_end_copy($conn);
?>
```

Aviso
Use of the **pg_put_line()** causes most large object operations, including **pg_lo_read()** and **pg_lo_tell()**, to subsequently fail. You can use **pg_copy_from()** and **pg_copy_to()** instead.

See also **pg_end_copy()**.

pg_query

(PHP 4 >= 4.2.0, PHP 5)

pg_query -- Execute a query

Description

resource **pg_query** (string query)

resource **pg_query** (resource connection, string query)

pg_query() returns a query result resource if query could be executed. It returns **FALSE** on failure or if connection is not a valid connection. Details about the error can be retrieved using the **pg_last_error()** function if connection is valid. **pg_query()** sends an SQL statement to the PostgreSQL database specified by the *connection* resource. The *connection* must be a valid connection that was returned by **pg_connect()** or **pg_pconnect()**. The return value of this function is an query result resource to be used to access the results from other PostgreSQL functions such as **pg_fetch_array()**.

> **Nota:** *connection* is an optional parameter for **pg_query()**. If *connection* is not set, default connection is used. Default connection is the last connection made by **pg_connect()** or **pg_pconnect()**.
>
> Although *connection* can be omitted, it is not recommended, since it could be a cause of hard to find bug in script.
>
> **Nota:** This function used to be called *pg_exec()*. *pg_exec()* is still available for compatibility reasons but users are encouraged to use the newer name.

See also **pg_connect()**, **pg_pconnect()**, **pg_fetch_array()**, **pg_fetch_object()**, **pg_num_rows()** and **pg_affected_rows()**.

pg_result_error

(PHP 4 >= 4.2.0, PHP 5)

pg_result_error -- Get error message associated with result

Description

string **pg_result_error** (resource result)

pg_result_error() returns error message associated with *result* resource. Therefore, user has better chance to get better error message than **pg_last_error()**.

Because **pg_query()** returns **FALSE** if the query fails, you must use **pg_send_query()** and **pg_get_result()** to get the result handle.

See also **pg_query()**, **pg_send_query()**, **pg_get_result()**, **pg_last_error()** and **pg_last_notice()**

pg_result_seek

(PHP 4 >= 4.3.0, PHP 5)

pg_result_seek -- Set internal row offset in result resource

Description

array **pg_result_seek** (resource result, int offset)

pg_result_seek() set internal row offset in result resource. It returns **FALSE**, if there is error.

See also **pg_fetch_row()**, **pg_fetch_assoc()**, **pg_fetch_array()**, **pg_fetch_object()** and **pg_fetch_result()**.

pg_result_status

(PHP 4 >= 4.2.0, PHP 5)

pg_result_status -- Get status of query result

Description

int **pg_result_status** (resource result)

pg_result_status() returns status of result resource. Possible return values are PGSQL_EMPTY_QUERY, PGSQL_COMMAND_OK, PGSQL_TUPLES_OK, PGSQL_COPY_TO, PGSQL_COPY_FROM, PGSQL_BAD_RESPONSE, PGSQL_NONFATAL_ERROR and PGSQL_FATAL_ERROR.

See also **pg_connection_status()**.

pg_select

(PHP 4 >= 4.3.0, PHP 5)

pg_select -- Select records

Description

array **pg_select** (resource connection, string table_name, array assoc_array [, int options])

pg_select() selects records specified by *assoc_array* which has *field=>value*. For successful query, it returns array contains all records and fields that match the condition specified by *assoc_array*. If *options* is specified, **pg_convert()** is applied to *assoc_array* with specified option.

Ejemplo 1. pg_select() example

```
<?php
    $db = pg_connect('dbname=foo');
    // This is safe, since $_POST is converted automatically
    $rec = pg_select($db, 'post_log', $_POST);
    if ($rec) {
        echo "Records selected\n";
        var_dump($rec);
    } else {
        echo "User must have sent wrong inputs\n";
    }
?>
```

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

See also **pg_convert()**

pg_send_query

(PHP 4 >= 4.2.0, PHP 5)

pg_send_query -- Sends asynchronous query

Description

bool **pg_send_query** (resource connection, string query)

bool **pg_send_query** (string query)

pg_send_query() send asynchronous query to the *connection*. Unlike **pg_query()**, it can send multiple query to PostgreSQL and get the result one by one using **pg_get_result()**. Script execution is not blocked while query is executing. Use **pg_connection_busy()** to check connection is busy (i.e. query is executing). Query may be cancelled by calling **pg_cancel_query()**.

Although user can send multiple query at once, user cannot send multiple query over busy connection. If query is sent while connection is busy, it waits until last query is finished and discards all result.

Ejemplo 1. Asynchronous Queries

```php
<?php
    $dbconn = pg_connect("dbname=publisher") or die("Could not connect");

    if (!pg_connection_busy($dbconn)) {
        pg_send_query($dbconn, "select * from authors; select count(*) from authors;");
    }

    $res1 = pg_get_result($dbconn);
    echo "First call to pg_get_result(): $res1\n";
    $rows1 = pg_num_rows($res1);
    echo "$res1 has $rows1 records\n\n";

    $res2 = pg_get_result($dbconn);
    echo "second call to pg_get_result(): $res2\n";
    $rows2 = pg_num_rows($res2);
    echo "$res2 has $rows2 records\n";
?>
```

The above example would produce the following output:

```
first call to pg_get_result(): Resource id #3
Resource id #3 has 3 records

second call to pg_get_result(): Resource id #4
Resource id #4 has 1 records
```

See also **pg_query()**, **pg_cancel_query()**, **pg_get_result()** and **pg_connection_busy()**.

pg_set_client_encoding

(PHP 3 CVS only, PHP 4 >= 4.0.3, PHP 5)

pg_set_client_encoding -- Set the client encoding

Description

int **pg_set_client_encoding** (string encoding)

int **pg_set_client_encoding** (resource connection, string encoding)

pg_set_client_encoding() sets the client encoding and returns 0 if success or -1 if error.

encoding is the client encoding and can be either : SQL_ASCII, EUC_JP, EUC_CN, EUC_KR, EUC_TW, UNICODE, MULE_INTERNAL, LATINX (X=1...9), KOI8, WIN, ALT, SJIS, BIG5, WIN1250. Available encoding depends on your PostgreSQL and libpq version. Refer to PostgreSQL manual for supported encodings for your PostgreSQL.

> **Nota:** This function requires PHP-4.0.3 or higher and PostgreSQL-7.0 or higher. Supported encoding depends on PostgreSQL version. Refer to PostgreSQL manual for details.
>
> The function used to be called *pg_setclientencoding()*.

See also **pg_client_encoding()**.

pg_trace

(PHP 4 >= 4.0.1, PHP 5)

pg_trace -- Enable tracing a PostgreSQL connection

Description

bool **pg_trace** (string pathname [, string mode [, resource connection]])

pg_trace() enables tracing of the PostgreSQL frontend/backend communication to a debugging file specified as *pathname*. To fully understand the results, one needs to be familiar with the internals of PostgreSQL communication protocol. For those who are not, it can still be useful for tracing errors in queries sent to the server, you could do for example **grep '^To backend' trace.log** and see what query actually were sent to the PostgreSQL server. For more information, refer to PostgreSQL manual.

pathname and *mode* are the same as in **fopen()** (*mode* defaults to 'w'), *connection* specifies the connection to trace and defaults to the last one opened.

pg_trace() returns **TRUE** if *pathname* could be opened for logging, **FALSE** otherwise.

See also **fopen()** and **pg_untrace()**.

pg_tty

(PHP 3, PHP 4 , PHP 5)

pg_tty -- Devuelve el nombre del tty

Descripción

string **pg_tty** (int connection_id)

pg_tty() devuelve el nombre del tty hacia el que se dirije la salida de depuración del lado del servidor en el identificador de conexión de PostgreSQL dado.

pg_unescape_bytea

(PHP 4 >= 4.3.0, PHP 5)

pg_unescape_bytea -- Unescape binary for bytea type

Description

string **pg_unescape_bytea** (string data)

pg_unescape_bytea() unescapes string from bytea datatype. It returns unescaped string (binary).

> **Nota:** When you SELECT bytea type, PostgreSQL returns octal byte value prefixed by \ (e.g. \032). Users are supposed to convert back to binary format by yourself.
>
> This function requires PostgreSQL 7.2 or later. With PostgreSQL 7.2.0 and 7.2.1, bytea type must be casted when you enable multi-byte support. i.e. *INSERT INTO test_table*

(image) VALUES ('$image_escaped'::bytea); PostgreSQL 7.2.2 or later does not need cast. Exception is when client and backend character encoding does not match, there may be multi-byte stream error. User must cast to bytea to avoid this error.

See also **pg_escape_bytea()** and **pg_escape_string()**

pg_untrace

(PHP 4 >= 4.0.1, PHP 5)

pg_untrace -- Disable tracing of a PostgreSQL connection

Description

bool **pg_untrace** ([resource connection])

Stop tracing started by **pg_trace()**. *connection* specifies the connection that was traced and defaults to the last one opened.

Returns always **TRUE**.

See also **pg_trace()**.

pg_update

(PHP 4 >= 4.3.0, PHP 5)

pg_update -- Update table

Description

mixed **pg_update** (resource connection, string table_name, array data, array condition [, int options])

pg_update() updates records that matches *condition* with *data*. If *options* is specified, **pg_convert()** is applied to *data* with specified options.

Ejemplo 1. pg_update() example

```
<?php
    $db = pg_connect('dbname=foo');
    $data = array('field1'=>'AA', 'field2'=>'BB');

    // This is safe, since $_POST is converted automatically
    $res = pg_update($db, 'post_log', $_POST, $data);
    if ($res) {
        echo "Data is updated: $res\n";
    } else {
        echo "User must have sent wrong inputs\n";
    }
?>
```

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

See also **pg_convert()**.

pg_version

(PHP 5)

pg_version -- Returns an array with client, protocol and server version (when available)

Description

array **pg_version** ([resource connection])

pg_version() returns an array with the client, protocol and server version. Protocol and server versions are only avaliable if PHP was compiled with PostgreSQL 7.4 or later.

CI. Funciones POSIX

Introducción

Este módulo contiene una interfaz con aquellas funciones definidas en el documento estándar IEEE 1003.1 (POSIX.1), y que no son asequibles de otra manera. POSIX.1, por ejemplo, definió también las funciones open(), read(), write() y close(), las cuales han sido tradicionalmente parte de PHP 3 durante mucho tiempo. Sin embargo, algunas funciones más específicas del sistema no habían estado disponibles antes, y éste módulo intenta remediar esto ofreciendo un acceso fácil a esas funciones.

Aviso
Pueden recuperarse datos sensibles con las funciones POSIX, p.ej. **posix_getpwnam()** y amigos. Ninguna de las funciones POSIX realizan algún tipo de chequeo de acceso cuando el modo seguro se encuentra habilitado. Por lo tanto, es **muy** recomendable deshabilitar la extensión POSIX por completo (use *--disable-posix* en su línea de configuración) si está operando en tal tipo de entorno.

Nota: Esta extensión no está disponible en plataformas Windows

---

Instalación

Las funciones POSIX son habilitadas por defecto. Puede deshabilitar las funciones tipo POSIX con *--disable-posix*.

Ver también

La sección sobre Funciones de Control de Procesos puede ser de su interés.

Tabla de contenidos

posix_access

(no version information, might be only in CVS)

posix_access -- Determine accessibility of a file

Descripción

bool **posix_access** (string file [, int mode])

posix_access() checks the user's permission of a file.

> **Nota:** Cuando safe-mode (modo-seguro) está activado, PHP comprueba si los archivos o directorios que va a utilizar tienen la misma UID que el script que está siendo ejecutado.

Lista de parámetros

file

The name of the file to be tested.

mode

A mask consisting of one or more of **POSIX_F_OK**, **POSIX_R_OK**, **POSIX_W_OK** and **POSIX_X_OK**. Defaults to **POSIX_F_OK**.

POSIX_R_OK, **POSIX_W_OK** and **POSIX_X_OK** request checking whether the file exists and has read, write and execute permissions, respectively. **POSIX_F_OK** just requests checking for the existence of the file.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplos

Ejemplo 1. posix_access() example

This example will check if the $file is readable and writable, otherwise will print an error message.

```
<?php

$file = 'some_file';

if (posix_access($file, POSIX_R_OK | POSIX_W_OK)) {
    echo 'The file is readable and writable!';

} else {
    $error = posix_get_last_error();

    echo "Error $error: " . posix_strerror($error);
}

?>
```

Ver también

posix_get_last_error()
posix_strerror()

posix_ctermid

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

posix_ctermid -- Recoge el nombre de ruta de la terminal de control

Descripción

string **posix_ctermid** (void)

Necesita ser escrito.

posix_get_last_error

(PHP 4 >= 4.2.0, PHP 5)

posix_get_last_error -- Recuperar el número de error establecido por la última función posix que ha fallado

Descripción

int **posix_get_last_error** (void)

Devuelve el valor errno (número de error) establecido por la última función posix que falló. Si no existe error, el valor 0 es devuelto. Si desea contar con el mensaje de error del sistema asociado con el errno, utilice **posix_strerror()**.

Vea también **posix_strerror()**.

posix_getcwd

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

posix_getcwd -- Nombre de ruta del directorio actual

Descripción

string **posix_getcwd** (void)

Necesita ser escrito cuanto antes.

posix_getegid

(PHP 3>= 3.0.10, PHP 4 , PHP 5)

posix_getegid -- Devuelve el ID de grupo efectivo del proceso actual

Descripción

int **posix_getegid** (void)

Devuelve el valor numérico ID de grupo efectivo del proceso actual. Vea también **posix_getgrgid()** para información sobre como convertir este número en un nombre de grupo manejable.

posix_geteuid

(PHP 3>= 3.0.10, PHP 4 , PHP 5)

posix_geteuid -- Devuelve el ID de usuario efectivo del proceso actual

Descripción

int **posix_geteuid** (void)

Devuelve el valor numérico ID de usuario efectivo del proceso actual. Vea también **posix_getpwuid()** para información sobre como convertir este número en un nombre de usuario manejable.

posix_getgid

(PHP 3>= 3.0.10, PHP 4 , PHP 5)

posix_getgid -- Devuelve el ID de grupo real del proceso actual

Descripción

int **posix_getgid** (void)

Devuelve el valor númerico ID de grupo real del proceso actual. Vea también **posix_getgrgid()** para información sobre como convertir esto en un nombre de grupo manejable.

posix_getgrgid

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

posix_getgrgid -- Devuelve información sobre un grupo a trave del id de grupo

Descripción

array **posix_getgrgid** (int gid)

Necesita ser escrito.

posix_getgrnam

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

posix_getgrnam -- Devuelve información sobre un grupo a traves del nombre

Descripción

array **posix_getgrnam** (string name)

Necesita ser escrito.

posix_getgroups

(PHP 3>= 3.0.10, PHP 4 , PHP 5)

posix_getgroups -- Devuelve el conjunto de grupos del proceso actual

Descripción

array **posix_getgroups** (void)

Devuelve un vector de enteros que contiene los ids numéricos de grupo de el conjunto de grupos del proceso actual. Vea también **posix_getgrgid()** para información sobre como convertir esto en nombres de grupo manejables.

posix_getlogin

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

posix_getlogin -- Devuelve el nombre de usuario

Descripción

string **posix_getlogin** (void)

Devuelve el nombre de usuario (login) que es dueño del proceso actual. Vea **posix_getpwnam()** para información sobre como conseguir mas datos de este usuario.

posix_getpgid

(PHP 3>= 3.0.10, PHP 4 , PHP 5)

posix_getpgid -- Recoge el id del grupo de procesos para el control de trabajo

Descripción

int **posix_getpgid** (int pid)

Devuelve el identificador de grupo de procesos del proceso *pid*.

Esta no es una función POSIX, pero es normal en sistemas BSD y System V. Si su sistema no soporta esta función a nivel de sistema, esta función PHP devolverá siempre **FALSE**.

posix_getpgrp

(PHP 3>= 3.0.10, PHP 4 , PHP 5)

posix_getpgrp -- Devuelve el identificador de grupo del proceso actual

Descripción

int **posix_getpgrp** (void)

Devuelve el identificador de grupo de proceso del proceso actual. Vea POSIX.1 y la página de

manual getpgrp(2) de su sistema POSIX para más información de grupos de procesos.

posix_getpid

(PHP 3>= 3.0.10, PHP 4 , PHP 5)

posix_getpid -- Devuelve el identificador del proceso actual

Descripción

int **posix_getpid** (void)

Devuelve el identificador de proceso del proceso actual.

posix_getppid

(PHP 3>= 3.0.10, PHP 4 , PHP 5)

posix_getppid -- Devuelve el identificador del proceso padre

Descripción

int **posix_getppid** (void)

Devuelve el identificador de proceso del proceso padre del proceso actual.

posix_getpwnam

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

posix_getpwnam -- Devuelve información sobre un usuario a traves del nombre de usuario

Description

array **posix_getpwnam** (string username)

Devuelve un vector asociativo conteniendo información sobre un usuario referenciado por un nombre alfanumérico, pasado a la función en el parametro *username*.

Los elementos del vector devuelto son:

Tabla 1. El vector de información de usuario

Elemento	Descripción
name	El elemento name contiene el nombre de usuario del usuario. Este es un nombre, normalmente menor de 16 caracteres, que no es su nombre completo, pero identifica al usario. Este debe ser el mismo que el parámetro *username* usado en la llamada a la función y por lo tanto es redundante.

Elemento	Descripción
passwd	El elemento passwd contiene la contraseña del usuario en un formato encriptado. Normalmente, por ejemplo en un sistema que este utilizando contraseñas "shadow", devolverá un asterisco.
uid	El ID de usuario del usuario en formato numérico.
gid	El ID de grupo del usuario. Utiliza la función **posix_getgrgid()** para resolver el nombre del grupo y una lista de sus miembros.
gecos	GECOS es un término obsoleto que se refiere al campo apuntado de información en un sistema de procesamiento batch Honeywell. El campo y sus contenidos han sido formalizado por POSIX y contiene una lista separada por comas con el nombre completo del usuario, teléfono del trabajo, número de oficina y télefono de casa. En muchos sistemas solo está disponible el nombre completo del usuario.
dir	Este elemento contiene la ruta absoluta al directorio del usuario (directorio home).
shell	El elemento shell contiene la ruta absoluta al ejecutable del shell por defecto del usuario.

posix_getpwuid

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

posix_getpwuid -- Devuelve información sobre un usuario a traves de su id

Descripción

array **posix_getpwuid** (int uid)

Devuelve un vector asociativo que contiene información sobre un usuario referenciado con un ID de usuario, pasado por el parámetro *uid*.

Los elementos del array son:

Tabla 1. El vector de información del usuario

Elemento	Descripción
name	El elemento name contiene el nombre de usuario del usuario. Este es un nombre, normalmente menor de 16 caracteres, que no es su verdadero nombre.
passwd	El elemento passwd contiene la contraseña del usuario en un formato encriptado. Normalmente, por ejemplo en un sistema con contraseñas "shadow", devolverá un asterisco.
uid	ID del usuario, debe ser el mismo que el parametro *uid* usado en la llamada a la función, y por lo tanto redundante.
gid	El ID del grupo del usuario. Utiliza la función **posix_getgrgid()** para resolver el nombre del grupo y una lista de sus miembros.

Elemento	Descripción
gecos	GECOS es un término obsoleto que se refiere al campo apuntado de de información en un sistema de procesamiento batch Honeywell. El campo y sus contenidos han sido formalizados por POSIX y contiene una lista separada por comas con el nombre completo del usuario, teléfono del trabajo, número de oficina y teléfono de casa. En muchos sistemas solo está disponible el nombre completo del usuario.
dir	Este elemento contiene la ruta absoluta al directorio del usuario (directorio home).
shell	El elemento shell contiene la ruta absoluta al ejecutable del shell por defecto del usuario.

posix_getrlimit

(PHP 3>= 3.0.10, PHP 4 , PHP 5)

posix_getrlimit -- Devuelve información sobre los limites de recursos del sistema

Descripción

array **posix_getrlimit** (void)

Necesita ser escrita tan pronto como sea posible.

posix_getsid

(PHP 3>= 3.0.10, PHP 4 , PHP 5)

posix_getsid -- Consigue el sid actual del proceso

Descripción

int **posix_getsid** (int pid)

Devuelve el sid del proceso *pid*. Si *pid* es 0, se devolverá el sid del proceso actual.

Esta no es una función POSIX, pero es normal en sistemas System V. Si su sistema no soporta esta función a nivel de sistema, esta función PHP devolverá siempre **`FALSE`**.

posix_getuid

(PHP 3>= 3.0.10, PHP 4 , PHP 5)

posix_getuid -- Devuelve el ID de usuario real del proceso actual

Descripción

int **posix_getuid** (void)

Devuelve el valor numerico ID de usuario real del proceso actual. Vea también **posix_getpwuid()**

para información sobre como convertir este ID en un nombre de usuario manejable.

posix_isatty

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

posix_isatty -- Determinar si un descriptor de archivo es una terminal interactiva

Descripción

bool **posix_isatty** (int da)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

posix_kill

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

posix_kill -- Manda una señal a un proceso

Descripción

bool **posix_kill** (int pid, int sig)

Manda la señal *sig* al proceso con el identificador de proceso *pid*. Devuelve **`FALSE`**, si no puede enviar la señal.Si sí la envia devuelve **`TRUE`** .

Vea también la página de manual kill(2) de su sistema POSIX, la cual contiene información adicional sobre los identificadores de proceso negativos, el pid especial 0, el pid especial -1, y la señal numero 0.

posix_mkfifo

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

posix_mkfifo -- Crear un archivo especial fifo (un pipe con nombre)

Descripción

bool **posix_mkfifo** (string nombre_ruta, int modo)

posix_mkfifo() crea un archivo *FIFO* especial que existe en el sistema de archivos y actúa como un punto de comunicación bi-direccional para los procesos.

El segundo parámetro *modo* tiene que ser definido en notación octal (p.ej. 0644). El permiso del *FIFO* recién creado depende también del valor **umask()** actual. Los permisos del archivo creado son (modo & ~umask).

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

> **Nota:** Cuando safe-mode (modo-seguro) está activado, PHP comprueba si los directorios que va a utilizar tienen la misma UID que el script que está siendo ejecutado.

posix_setegid

(PHP 4 >= 4.0.2, PHP 5)

posix_setegid -- Establecer el GID efectivo del proceso actual

Descripción

bool **posix_setegid** (int gid)

Establece el ID de grupo efectivo del proceso actual. Esta es una función privilegiada y usted necesita los permisos apropiados (usualmente root) en su sistema para contar con la capacidad de ejecutar esta función.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

posix_seteuid

(PHP 4 >= 4.0.2, PHP 5)

posix_seteuid -- Establecer el UID efectivo del proceso actual

Descripción

bool **posix_seteuid** (int uid)

Establece el ID de usuario real del proceso actual. Esta es una función privilegiada y usted necesita los permisos apropiados (usualmente root) en su sistema para tener la capacidad de ejecutar esta función.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Vea también **posix_setgid()**.

posix_setgid

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

posix_setgid -- Asigna el GID efectivo del proceso actual

Descripción

bool **posix_setgid** (int gid)

Asigna el ID de grupo real del proceso actual. Esta es una función privilegiada y necesitas los privilegion apropiados (normalmente root) en tu sistema para realizar esta función. El orden apropiado de llamada es **posix_setgid()** primero, **posix_setuid()** después.

Devuelve **TRUE** si tiene éxito, **FALSE** en caso contrario.

posix_setpgid

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

posix_setpgid -- Asigna el id de grupo de procesos para el control de trabajos

Descripción

int **posix_setpgid** (int pid, int pgid)

Inserta el proceso *pid* en el grupo de procesos *pgid*. Vea POSIX.1 y la página de manual setsid(2) de su sistema POSIX para más información sobre grupos de procesos y control de trabajo. Devuelve **TRUE** si tiene éxito y **FALSE** en caso contrario.

posix_setsid

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

posix_setsid -- Convierte el proceso actual en lider de sesión

Descripción

int **posix_setsid** (void)

Convierte el proceso actual en lider de sesión. Vea POSIX.1 y la página de manual setsid(2) en su sistema POSIX para más informacion en grupos de proceso y control de trabajos. Devuelve el id de sesión.

posix_setuid

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

posix_setuid -- Asigna el UID efectivo del proceso actual

Descripción

bool **posix_setuid** (int uid)

Asigna el ID de usuario real al proceso actual. Esta es una función privilegiada y necesitas los privilegios apropiados (normalmente root) en tu sistema para realizar esta función.

Devuelve **TRUE** si tiene exito, **FALSE** en caso contrario. Vea también **posix_setgid()**.

posix_strerror

(PHP 4 >= 4.2.0, PHP 5)

posix_strerror -- Recuperar el mensaje de error del sistema asociado con el errno dado

Descripción

string **posix_strerror** (int errno)

Devuelve el mensaje de error del sistema POSIX asociado con el errno dado. Si *errno* es 0, entonces la cadena "Success" es devuelta. La función **posix_get_last_error()** es usada para recuperar el último errno POSIX.

Vea también **posix_get_last_error()**.

posix_times

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

posix_times -- Recoge el tiempo de los procesos

Descripción

array **posix_times** (void)

Devuelve un hash de cadenas con información sobre el uso de CPU del proceso actual. Los índices del hash son

- ticks - el numero de ticks de reloj que han pasado desde el reinicio.
- utime - tiempo de usuario usado por el proceso actual.
- stime - tiempo de sistema usado por el proceso actual.
- cutime - tiempo de usuario usado por el proceso actual e hijos.
- cstime - tiempo de sistema usado por el proceso actual e hijos.

posix_ttyname

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

posix_ttyname -- Determina el nombre del dispositivo terminal

Descripción

string **posix_ttyname** (int fd)

Necesita ser escrito.

posix_uname

(PHP 3>= 3.0.10, PHP 4 , PHP 5)

posix_uname -- Consigue el nombre del sistema

Descripción

array **posix_uname** (void)

Devuelve un hash de cadenas con información sobre el sistema. Los índices del hash son

- sysname - nombre del sistema operativo (e.g. Linux)
- nodename - nombre del sistema (e.g. valiant)
- release - release del sistema operativo (e.g. 2.2.10)
- version - versión del sistema operativo (e.g. #4 Tue Jul 20 17:01:36 MEST 1999)
- machine - arquitectura del sistema (e.g. i586)

Posix requiere que usted no debe hacer ninguna suposición sobre el formato de los valores, por ejemplo usted no puede confiar en los tres dígitos de la version o cualquier cosa devuelta por esta función.

CII. Printer Functions

Introducción

These functions are only available under Windows 9.x, ME, NT4 and 2000. They have been added in PHP 4.0.4.

Instalación

This PECL extension is not bundled with PHP.

Windows users must enable `php_printer.dll` inside of `php.ini` in order to use these functions. You may obtain this unbundled PECL extension from the various PECL snaps pages (select the appropriate repository for your version of PHP): PECL for PHP 4.3.x, PECL for PHP 5.0.x or PECL Unstable.

Configuración en tiempo de ejecución

El comportamiento de estas funciones está afectado por los valores definidos en `php.ini`.

Tabla 1. Printer configuration options

Name	Default	Changeable
printer.default_printer	""	PHP_INI_ALL

For further details and definitions of the PHP_INI_* constants, see the Apéndice H.

Tabla de contenidos

printer_abort

(no version information, might be only in CVS)

printer_abort -- Deletes the printer's spool file

Description

void **printer_abort** (resource handle)

This function deletes the printers spool file.

handle must be a valid handle to a printer.

Ejemplo 1. printer_abort() example

```
<?php
$handle = printer_open();
printer_abort($handle);
printer_close($handle);
?>
```

printer_close

(no version information, might be only in CVS)

printer_close -- Close an open printer connection

Description

void **printer_close** (resource handle)

This function closes the printer connection. **printer_close()** also closes the active device context.

handle must be a valid handle to a printer.

Ejemplo 1. printer_close() example

```
<?php
$handle = printer_open();
printer_close($handle);
?>
```

printer_create_brush

(no version information, might be only in CVS)

printer_create_brush -- Create a new brush

Description

mixed **printer_create_brush** (int style, string color)

The function creates a new brush and returns a handle to it. A brush is used to fill shapes. For an example see **printer_select_brush()**. *color* must be a color in RGB hex format, i.e. "000000" for black, *style* must be one of the following constants:

- *PRINTER_BRUSH_SOLID*: creates a brush with a solid color.
- *PRINTER_BRUSH_DIAGONAL*: creates a brush with a 45-degree upward left-to-right hatch (/).
- *PRINTER_BRUSH_CROSS*: creates a brush with a cross hatch (+).
- *PRINTER_BRUSH_DIAGCROSS*: creates a brush with a 45 cross hatch (x).
- *PRINTER_BRUSH_FDIAGONAL*: creates a brush with a 45-degree downward left-to-right hatch (\ ).

- *PRINTER_BRUSH_HORIZONTAL*: creates a brush with a horizontal hatch (-).
- *PRINTER_BRUSH_VERTICAL*: creates a brush with a vertical hatch (|).
- *PRINTER_BRUSH_CUSTOM*: creates a custom brush from an BMP file. The second parameter is used to specify the BMP instead of the RGB color code.

printer_create_dc

(no version information, might be only in CVS)

printer_create_dc -- Create a new device context

Description

void **printer_create_dc** (resource handle)

The function creates a new device context. A device context is used to customize the graphic objects of the document. *handle* must be a valid handle to a printer.

Ejemplo 1. printer_create_dc() example

```
<?php
$handle = printer_open();
printer_start_doc($handle);
printer_start_page($handle);

printer_create_dc($handle);
/* do some stuff with the dc */
printer_set_option($handle, PRINTER_TEXT_COLOR, "333333");
printer_draw_text($handle, 1, 1, "text");
printer_delete_dc($handle);

/* create another dc */
printer_create_dc($handle);
printer_set_option($handle, PRINTER_TEXT_COLOR, "000000");
printer_draw_text($handle, 1, 1, "text");
/* do some stuff with the dc */

printer_delete_dc($handle);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
?>
```

printer_create_font

(no version information, might be only in CVS)

printer_create_font -- Create a new font

Description

mixed **printer_create_font** (string face, int height, int width, int font_weight, bool italic, bool underline, bool strikeout, int orientation)

The function creates a new font and returns a handle to it. A font is used to draw text. For an example see **printer_select_font()**. *face* must be a string specifying the font face. *height* specifies the font height, and *width* the font width. The *font_weight* specifies the font weight (400 is normal), and can be one of the following predefined constants.

- *PRINTER_FW_THIN*: sets the font weight to thin (100).
- *PRINTER_FW_ULTRALIGHT*: sets the font weight to ultra light (200).
- *PRINTER_FW_LIGHT*: sets the font weight to light (300).
- *PRINTER_FW_NORMAL*: sets the font weight to normal (400).
- *PRINTER_FW_MEDIUM*: sets the font weight to medium (500).
- *PRINTER_FW_BOLD*: sets the font weight to bold (700).
- *PRINTER_FW_ULTRABOLD*: sets the font weight to ultra bold (800).
- *PRINTER_FW_HEAVY*: sets the font weight to heavy (900).

italic can be **TRUE** or **FALSE**, and sets whether the font should be italic.

underline can be **TRUE** or **FALSE**, and sets whether the font should be underlined.

strikeout can be **TRUE** or **FALSE**, and sets whether the font should be stroked out.

orientation specifies a rotation. For an example see **printer_select_font()**.

printer_create_pen

(no version information, might be only in CVS)

printer_create_pen -- Create a new pen

Description

mixed **printer_create_pen** (int style, int width, string color)

The function creates a new pen and returns a handle to it. A pen is used to draw lines and curves. For an example see **printer_select_pen()**. *color* must be a color in RGB hex format, i.e. "000000" for black, *width* specifies the width of the pen whereas *style* must be one of the following constants:

- *PRINTER_PEN_SOLID*: creates a solid pen.
- *PRINTER_PEN_DASH*: creates a dashed pen.
- *PRINTER_PEN_DOT*: creates a dotted pen.
- *PRINTER_PEN_DASHDOT*: creates a pen with dashes and dots.
- *PRINTER_PEN_DASHDOTDOT*: creates a pen with dashes and double dots.

- *PRINTER_PEN_INVISIBLE*: creates an invisible pen.

printer_delete_brush

(no version information, might be only in CVS)

printer_delete_brush -- Delete a brush

Description

bool **printer_delete_brush** (resource handle)

The function deletes the selected brush. For an example see **printer_select_brush()**. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo. *handle* must be a valid handle to a brush.

printer_delete_dc

(no version information, might be only in CVS)

printer_delete_dc -- Delete a device context

Description

bool **printer_delete_dc** (resource handle)

The function deletes the device context. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo. For an example see **printer_create_dc()**. *handle* must be a valid handle to a printer.

printer_delete_font

(no version information, might be only in CVS)

printer_delete_font -- Delete a font

Description

bool **printer_delete_font** (resource handle)

The function deletes the selected font. For an example see **printer_select_font()**. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo. *handle* must be a valid handle to a font.

printer_delete_pen

(no version information, might be only in CVS)

printer_delete_pen -- Delete a pen

Description

bool **printer_delete_pen** (resource handle)

The function deletes the selected pen. For an example see **printer_select_pen()**. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo. *handle* must be a valid handle to a pen.

printer_draw_bmp

(no version information, might be only in CVS)

printer_draw_bmp -- Draw a bmp

Description

void **printer_draw_bmp** (resource handle, string filename, int x, int y [, int width, int height])

The function simply draws an bmp the bitmap *filename* at position *x*, *y*. *handle* must be a valid handle to a printer.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplo 1. printer_draw_bmp() example

```
<?php
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

printer_draw_bmp($handle, "c:\\image.bmp", 1, 1);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
?>
```

printer_draw_chord

(no version information, might be only in CVS)

printer_draw_chord -- Draw a chord

Description

void **printer_draw_chord** (resource handle, int rec_x, int rec_y, int rec_x1, int rec_y1, int rad_x, int rad_y, int rad_x1, int rad_y1)

The function simply draws an chord. *handle* must be a valid handle to a printer.

rec_x is the upper left x coordinate of the bounding rectangle.

rec_y is the upper left y coordinate of the bounding rectangle.

rec_x1 is the lower right x coordinate of the bounding rectangle.

rec_y1 is the lower right y coordinate of the bounding rectangle.

rad_x is x coordinate of the radial defining the beginning of the chord.

rad_y is y coordinate of the radial defining the beginning of the chord.

rad_x1 is x coordinate of the radial defining the end of the chord.

rad_y1 is y coordinate of the radial defining the end of the chord.

Ejemplo 1. printer_draw_chord() example

```
<?php
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_chord($handle, 1, 1, 500, 500, 1, 1, 500, 1);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
?>
```

printer_draw_elipse

(no version information, might be only in CVS)

printer_draw_elipse -- Draw an ellipse

Description

void **printer_draw_elipse** (resource handle, int ul_x, int ul_y, int lr_x, int lr_y)

The function simply draws an ellipse. *handle* must be a valid handle to a printer.

ul_x is the upper left x coordinate of the ellipse.

ul_y is the upper left y coordinate of the ellipse.

lr_x is the lower right x coordinate of the ellipse.

lr_y is the lower right y coordinate of the ellipse.

Ejemplo 1. printer_draw_elipse() example

```
<?php
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_elipse($handle, 1, 1, 500, 500);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
?>
```

printer_draw_line

(no version information, might be only in CVS)

printer_draw_line -- Draw a line

Description

void **printer_draw_line** (resource printer_handle, int from_x, int from_y, int to_x, int to_y)

The function simply draws a line from position *from_x*, *from_y* to position *to_x*, *to_y* using the selected pen. *printer_handle* must be a valid handle to a printer.

Ejemplo 1. printer_draw_line() example

```
<?php
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 30, "000000");
printer_select_pen($handle, $pen);

printer_draw_line($handle, 1, 10, 1000, 10);
printer_draw_line($handle, 1, 60, 500, 60);

printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
?>
```

printer_draw_pie

(no version information, might be only in CVS)

printer_draw_pie -- Draw a pie

Description

void **printer_draw_pie** (resource handle, int rec_x, int rec_y, int rec_x1, int rec_y1, int rad1_x, int rad1_y, int rad2_x, int rad2_y)

The function simply draws an pie. *handle* must be a valid handle to a printer.

rec_x is the upper left x coordinate of the bounding rectangle.

rec_y is the upper left y coordinate of the bounding rectangle.

rec_x1 is the lower right x coordinate of the bounding rectangle.

rec_y1 is the lower right y coordinate of the bounding rectangle.

rad1_x is x coordinate of the first radial's ending.

rad1_y is y coordinate of the first radial's ending.

rad2_x is x coordinate of the second radial's ending.

rad2_y is y coordinate of the second radial's ending.

Ejemplo 1. printer_draw_pie() example

```
<?php
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_pie($handle, 1, 1, 500, 500, 1, 1, 500, 1);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
?>
```

printer_draw_rectangle

(no version information, might be only in CVS)

printer_draw_rectangle -- Draw a rectangle

Description

void **printer_draw_rectangle** (resource handle, int ul_x, int ul_y, int lr_x, int lr_y)

The function simply draws a rectangle.

handle must be a valid handle to a printer.

ul_x is the upper left x coordinate of the rectangle.

ul_y is the upper left y coordinate of the rectangle.

lr_x is the lower right x coordinate of the rectangle.

lr_y is the lower right y coordinate of the rectangle.

Ejemplo 1. printer_draw_rectangle() example

```
<?php
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_rectangle($handle, 1, 1, 500, 500);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
?>
```

printer_draw_roundrect

(no version information, might be only in CVS)

printer_draw_roundrect -- Draw a rectangle with rounded corners

Description

void **printer_draw_roundrect** (resource handle, int ul_x, int ul_y, int lr_x, int lr_y, int width, int height)

The function simply draws a rectangle with rounded corners.

handle must be a valid handle to a printer.

ul_x is the upper left x coordinate of the rectangle.

ul_y is the upper left y coordinate of the rectangle.

lr_x is the lower right x coordinate of the rectangle.

lr_y is the lower right y coordinate of the rectangle.

width is the width of the ellipse.

height is the height of the ellipse.

Ejemplo 1. printer_draw_roundrect() example

```
<?php
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_roundrect($handle, 1, 1, 500, 500, 200, 200);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
?>
```

printer_draw_text

(no version information, might be only in CVS)

printer_draw_text -- Draw text

Description

void **printer_draw_text** (resource printer_handle, string text, int x, int y)

The function simply draws *text* at position *x*, *y* using the selected font. *printer_handle* must be a valid handle to a printer.

Ejemplo 1. printer_draw_text() example

```
<?php
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$font = printer_create_font("Arial", 72, 48, 400, false, false, false, 0);
printer_select_font($handle, $font);
printer_draw_text($handle, "test", 10, 10);
printer_delete_font($font);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
?>
```

printer_end_doc

(no version information, might be only in CVS)

printer_end_doc -- Close document

Description

bool **printer_end_doc** (resource handle)

Closes a new document in the printer spooler. The document is now ready for printing. For an example see **printer_start_doc()**. *handle* must be a valid handle to a printer.

printer_end_page

(no version information, might be only in CVS)

printer_end_page -- Close active page

Description

bool **printer_end_page** (resource handle)

The function closes the active page in the active document. For an example see **printer_start_doc()**. *handle* must be a valid handle to a printer.

printer_get_option

(no version information, might be only in CVS)

printer_get_option -- Retrieve printer configuration data

Description

mixed **printer_get_option** (resource handle, string option)

The function retrieves the configuration setting of *option*. *handle* must be a valid handle to a printer. Take a look at **printer_set_option()** for the settings that can be retrieved, additionally the following settings can be retrieved:

- *PRINTER_DEVICENAME* returns the devicename of the printer.
- *PRINTER_DRIVERVERSION* returns the printer driver version.

Ejemplo 1. printer_get_option() example

```
<?php
$handle = printer_open();
echo printer_get_option($handle, PRINTER_DRIVERVERSION);
printer_close($handle);
?>
```

printer_list

(no version information, might be only in CVS)

printer_list -- Return an array of printers attached to the server

Description

array **printer_list** (int enumtype [, string name [, int level]])

The function enumerates available printers and their capabilities. *level* sets the level of information request. Can be 1,2,4 or 5. *enumtype* must be one of the following predefined constants:

- *PRINTER_ENUM_LOCAL*: enumerates the locally installed printers.
- *PRINTER_ENUM_NAME*: enumerates the printer of *name*, can be a server, domain or print provider.
- *PRINTER_ENUM_SHARED*: this parameter can't be used alone, it has to be OR'ed with other parameters, i.e. PRINTER_ENUM_LOCAL to detect the locally shared printers.
- *PRINTER_ENUM_DEFAULT*: (Win9.x only) enumerates the default printer.
- *PRINTER_ENUM_CONNECTIONS*: (WinNT/2000 only) enumerates the printers to which the user has made connections.
- *PRINTER_ENUM_NETWORK*: (WinNT/2000 only) enumerates network printers in the computer's domain. Only valid if *level* is 1.
- *PRINTER_ENUM_REMOTE*: (WinNT/2000 only) enumerates network printers and print servers in the computer's domain. Only valid if *level* is 1.

Ejemplo 1. printer_list() example

```
<?php
/* detect locally shared printer */
var_dump(printer_list(PRINTER_ENUM_LOCAL | PRINTER_ENUM_SHARED));
?>
```

printer_logical_fontheight

(no version information, might be only in CVS)

printer_logical_fontheight -- Get logical font height

Description

int **printer_logical_fontheight** (resource handle, int height)

The function calculates the logical font height of *height*. *handle* must be a valid handle to a printer.

Ejemplo 1. printer_logical_fontheight() example

```
<?php
$handle = printer_open();
echo printer_logical_fontheight($handle, 72);
printer_close($handle);
?>
```

printer_open

(no version information, might be only in CVS)

printer_open -- Open connection to a printer

Description

mixed **printer_open** ([string devicename])

This function tries to open a connection to the printer *devicename*, and returns a handle on success or **FALSE** on failure.

If no parameter was given it tries to open a connection to the default printer (if not specified in `php.ini` as *printer.default_printer*, PHP tries to detect it).

printer_open() also starts a device context.

Ejemplo 1. printer_open() example

```
<?php
$handle = printer_open("HP Deskjet 930c");
$handle = printer_open();
?>
```

printer_select_brush

(no version information, might be only in CVS)

printer_select_brush -- Select a brush

Description

void **printer_select_brush** (resource printer_handle, resource brush_handle)

The function selects a brush as the active drawing object of the actual device context. A brush is used to fill shapes. If you draw an rectangle the brush is used to draw the shapes, while the pen is used to draw the border. If you haven't selected a brush before drawing shapes, the shape won't be filled. *printer_handle* must be a valid handle to a printer. *brush_handle* must be a valid handle to a brush.

Ejemplo 1. printer_select_brush() example

```
<?php
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);
$brush = printer_create_brush(PRINTER_BRUSH_CUSTOM, "c:\\brush.bmp");
printer_select_brush($handle, $brush);

printer_draw_rectangle($handle, 1, 1, 500, 500);

printer_delete_brush($brush);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "000000");
printer_select_brush($handle, $brush);
printer_draw_rectangle($handle, 1, 501, 500, 1001);
printer_delete_brush($brush);

printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
?>
```

printer_select_font

(no version information, might be only in CVS)

printer_select_font -- Select a font

Description

void **printer_select_font** (resource printer_handle, resource font_handle)

The function selects a font to draw text. *printer_handle* must be a valid handle to a printer. *font_handle* must be a valid handle to a font.

Ejemplo 1. printer_select_font() example

```
<?php
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$font = printer_create_font("Arial", 148, 76, PRINTER_FW_MEDIUM, false, false, false, -
printer_select_font($handle, $font);
printer_draw_text($handle, "PHP is simply cool", 40, 40);
printer_delete_font($font);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
?>
```

printer_select_pen

(no version information, might be only in CVS)

printer_select_pen -- Select a pen

Description

void **printer_select_pen** (resource printer_handle, resource pen_handle)

The function selects a pen as the active drawing object of the actual device context. A pen is used to draw lines and curves. I.e. if you draw a single line the pen is used. If you draw an rectangle the pen is used to draw the borders, while the brush is used to fill the shape. If you haven't selected a pen before drawing shapes, the shape won't be outlined. *printer_handle* must be a valid handle to a printer. *pen_handle* must be a valid handle to a pen.

Ejemplo 1. printer_select_pen() example

```
<?php
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 30, "2222FF");
printer_select_pen($handle, $pen);

printer_draw_line($handle, 1, 60, 500, 60);

printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
?>
```

printer_set_option

(no version information, might be only in CVS)

printer_set_option -- Configure the printer connection

Description

bool **printer_set_option** (resource handle, int option, mixed value)

The function sets the following options for the current connection. *handle* must be a valid handle to a printer. For *option* can be one of the following constants:

- *PRINTER_COPIES*: sets how many copies should be printed, *value* must be an integer.
- *PRINTER_MODE*: specifies the type of data (text, raw or emf), *value* must be a string.
- *PRINTER_TITLE*: specifies the name of the document, *value* must be a string.
- *PRINTER_ORIENTATION*: specifies the orientation of the paper, *value* can be either PRINTER_ORIENTATION_PORTRAIT or PRINTER_ORIENTATION_LANDSCAPE
- *PRINTER_RESOLUTION_Y*: specifies the y-resolution in DPI, *value* must be an integer.
- *PRINTER_RESOLUTION_X*: specifies the x-resolution in DPI, *value* must be an integer.
- *PRINTER_PAPER_FORMAT*: specifies the a predefined paper format, set *value* to

PRINTER_FORMAT_CUSTOM if you want to specify a custom format with PRINTER_PAPER_WIDTH and PRINTER_PAPER_LENGTH. *value* can be one of the following constants.

  - *PRINTER_FORMAT_CUSTOM*: let's you specify a custom paper format.
  - *PRINTER_FORMAT_LETTER*: specifies standard letter format (8 1/2- by 11-inches).
  - *PRINTER_FORMAT_LETTER*: specifies standard legal format (8 1/2- by 14-inches).
  - *PRINTER_FORMAT_A3*: specifies standard A3 format (297- by 420-millimeters).
  - *PRINTER_FORMAT_A4*: specifies standard A4 format (210- by 297-millimeters).
  - *PRINTER_FORMAT_A5*: specifies standard A5 format (148- by 210-millimeters).
  - *PRINTER_FORMAT_B4*: specifies standard B4 format (250- by 354-millimeters).
  - *PRINTER_FORMAT_B5*: specifies standard B5 format (182- by 257-millimeter).
  - *PRINTER_FORMAT_FOLIO*: specifies standard FOLIO format (8 1/2- by 13-inch).

- *PRINTER_PAPER_LENGTH*: if PRINTER_PAPER_FORMAT is set to PRINTER_FORMAT_CUSTOM, PRINTER_PAPER_LENGTH specifies a custom paper length in mm, *value* must be an integer.
- *PRINTER_PAPER_WIDTH*: if PRINTER_PAPER_FORMAT is set to PRINTER_FORMAT_CUSTOM, PRINTER_PAPER_WIDTH specifies a custom paper width in mm, *value* must be an integer.
- *PRINTER_SCALE*: specifies the factor by which the printed output is to be scaled. the page size is scaled from the physical page size by a factor of scale/100. for example if you set the scale to 50, the output would be half of its original size. *value* must be an integer.
- *PRINTER_BACKGROUND_COLOR*: specifies the background color for the actual device context, *value* must be a string containing the rgb information in hex format i.e. "005533".
- *PRINTER_TEXT_COLOR*: specifies the text color for the actual device context, *value* must be a string containing the rgb information in hex format i.e. "005533".
- *PRINTER_TEXT_ALIGN*: specifies the text alignment for the actual device context, *value* can be combined through OR'ing the following constants:
  - *PRINTER_TA_BASELINE*: text will be aligned at the base line.
  - *PRINTER_TA_BOTTOM*: text will be aligned at the bottom.
  - *PRINTER_TA_TOP*: text will be aligned at the top.
  - *PRINTER_TA_CENTER*: text will be aligned at the center.
  - *PRINTER_TA_LEFT*: text will be aligned at the left.
  - *PRINTER_TA_RIGHT*: text will be aligned at the right.

Ejemplo 1. printer_set_option() example

```
<?php
$handle = printer_open();
printer_set_option($handle, PRINTER_SCALE, 75);
printer_set_option($handle, PRINTER_TEXT_ALIGN, PRINTER_TA_LEFT);
printer_close($handle);
?>
```

printer_start_doc

(no version information, might be only in CVS)

printer_start_doc -- Start a new document

Description

bool **printer_start_doc** (resource handle [, string document])

The function creates a new document in the printer spooler. A document can contain multiple pages, it's used to schedule the print job in the spooler. *handle* must be a valid handle to a printer. The optional parameter *document* can be used to set an alternative document name.

Ejemplo 1. printer_start_doc() example

```
<?php
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
?>
```

printer_start_page

(no version information, might be only in CVS)

printer_start_page -- Start a new page

Description

bool **printer_start_page** (resource handle)

The function creates a new page in the active document. For an example see **printer_start_doc()**. *handle* must be a valid handle to a printer.

printer_write

(no version information, might be only in CVS)

printer_write -- Write data to the printer

Description

bool **printer_write** (resource handle, string content)

Writes *content* directly to the printer. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

handle must be a valid handle to a printer.

Ejemplo 1. printer_write() example

```
<?php
$handle = printer_open();
printer_write($handle, "Text to print");
printer_close($handle);
?>
```

CIII. Pspell Functions

The **pspell()** functions allow you to check the spelling of a word and offer suggestions.

You need the aspell and pspell libraries, available from http://aspell.sourceforge.net/ and http://aspell.net/ respectively, and add the *--with-pspell[=dir]* option when compiling php.

Tabla de contenidos

pspell_add_to_personal

(PHP 4 >= 4.0.2, PHP 5)

pspell_add_to_personal -- Agrega la palabra a la lista personal de palabras

Descripción

int **pspell_add_to_personal** (int dictionary_link, cadena palabra)

pspell_add_to_personal() agrega una palabra a la lista personal de palabras. Si usas **pspell_new_config()** con **pspell_config_personal()** para abrir el diccionario, puedes salvar la lista de palabras luego, con **pspell_save_wordlist()**. Nota: Esta función no funcionará hasta que tengas pspell .11.2 y aspell .32.5 o superior.

Ejemplo 1. pspell_add_to_personal()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
$pspell_link = pspell_new_config ($pspell_config);

pspell_add_to_personal ($pspell_link, "Vlad");
pspell_save_wordlist ($pspell_link);
```

pspell_add_to_session

(PHP 4 >= 4.0.2, PHP 5)

pspell_add_to_session -- Agrega la palabra a la lista de palabras en la sesión actual

Descripción

int **pspell_add_to_session** (int dictionary_link, cadena palabra)

pspell_add_to_session() agrega la palabra a la lista de palabras asociada con la sesión actual. Esto es muy similar a **pspell_add_to_personal()**

pspell_check

(PHP 4 >= 4.0.2, PHP 5)

pspell_check -- Check a word

Description

bool **pspell_check** (int dictionary_link, string word)

pspell_check() checks the spelling of a word and returns **TRUE** if the spelling is correct, **FALSE** if not.

Ejemplo 1. pspell_check()

```
<?php
$pspell_link = pspell_new("en");

if (pspell_check($pspell_link, "testt")) {
    echo "This is a valid spelling";
} else {
    echo "Sorry, wrong spelling";
}
?>
```

pspell_clear_session

(PHP 4 >= 4.0.2, PHP 5)

pspell_clear_session -- Limpia la sesión actual

Descripción

int **pspell_clear_session** (int dictionary_link)

pspell_clear_session() limpia la sesión actual. La lista de palabras actual queda vacía, y, por ejemplo, si intentas salvarla con **pspell_save_wordlist()**, nada ocurre.

Ejemplo 1. pspell_add_to_personal()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
$pspell_link = pspell_new_config ($pspell_config);

pspell_add_to_personal ($pspell_link, "Vlad");
pspell_clear_session ($pspell_link);
pspell_save_wordlist ($pspell_link);    //"Vlad" will not be saved
```

pspell_config_create

(PHP 4 >= 4.0.2, PHP 5)

pspell_config_create -- Create a config used to open a dictionary

Description

int **pspell_config_create** (string language [, string spelling [, string jargon [, string encoding]]])

pspell_config_create() has a very similar syntax to **pspell_new()**. In fact, using **pspell_config_create()** immediately followed by **pspell_new_config()** will produce the exact same result. However, after creating a new config, you can also use **pspell_config_*()** functions before calling **pspell_new_config()** to take advantage of some advanced functionality.

The language parameter is the language code which consists of the two letter ISO 639 language code and an optional two letter ISO 3166 country code after a dash or underscore.

The spelling parameter is the requested spelling for languages with more than one spelling such as English. Known values are 'american', 'british', and 'canadian'.

The jargon parameter contains extra information to distinguish two different words lists that have the same language and spelling parameters.

The encoding parameter is the encoding that words are expected to be in. Valid values are 'utf-8', 'iso8859-*', 'koi8-r', 'viscii', 'cp1252', 'machine unsigned 16', 'machine unsigned 32'. This parameter is largely untested, so be careful when using.

The mode parameter is the mode in which spellchecker will work. There are several modes available:

- **PSPELL_FAST** - Fast mode (least number of suggestions)
- **PSPELL_NORMAL** - Normal mode (more suggestions)
- **PSPELL_BAD_SPELLERS** - Slow mode (a lot of suggestions)

For more information and examples, check out inline manual pspell website:http://aspell.net/.

Ejemplo 1. pspell_config_create()

```
<?php
$pspell_config = pspell_config_create("en");
pspell_config_personal($pspell_config, "/var/dictionaries/custom.pws");
pspell_config_repl($pspell_config, "/var/dictionaries/custom.repl");
$pspell_link = pspell_new_personal($pspell_config, "en");
?>
```

pspell_config_data_dir

(PHP 5)

pspell_config_data_dir -- location of language data files

Description

bool **pspell_config_data_dir** (int conf, string directory)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

pspell_config_dict_dir

(PHP 5)

pspell_config_dict_dir -- Location of the main word list

Description

bool **pspell_config_dict_dir** (int conf, string directory)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

pspell_config_ignore

(PHP 4 >= 4.0.2, PHP 5)

pspell_config_ignore -- Ignore words less than N characters long

Description

int **pspell_config_ignore** (int dictionary_link, int n)

pspell_config_ignore() should be used on a config before calling **pspell_new_config()**. This function allows short words to be skipped by the spell checker. Words less than *n* characters will be skipped.

Ejemplo 1. pspell_config_ignore()

```
<?php
$pspell_config = pspell_config_create("en");
pspell_config_ignore($pspell_config, 5);
$pspell_link = pspell_new_config($pspell_config);
pspell_check($pspell_link, "abcd");    //will not result in an error
?>
```

pspell_config_mode

(PHP 4 >= 4.0.2, PHP 5)

pspell_config_mode -- Change the mode number of suggestions returned

Description

int **pspell_config_mode** (int dictionary_link, int mode)

pspell_config_mode() should be used on a config before calling **pspell_new_config()**. This function determines how many suggestions will be returned by **pspell_suggest()**.

The mode parameter is the mode in which spellchecker will work. There are several modes available:

- **PSPELL_FAST** - Fast mode (least number of suggestions)
- **PSPELL_NORMAL** - Normal mode (more suggestions)
- **PSPELL_BAD_SPELLERS** - Slow mode (a lot of suggestions)

Ejemplo 1. pspell_config_mode()

```
<?php
$pspell_config = pspell_config_create("en");
pspell_config_mode($pspell_config, PSPELL_FAST);
$pspell_link = pspell_new_config($pspell_config);
pspell_check($pspell_link, "thecat");
?>
```

pspell_config_personal

(PHP 4 >= 4.0.2, PHP 5)

pspell_config_personal -- Set a file that contains personal wordlist

Description

int **pspell_config_personal** (int dictionary_link, string file)

pspell_config_personal() should be used on a config before calling **pspell_new_config()**. The personal wordlist will be loaded and used in addition to the standard one after you call **pspell_new_config()**. If the file does not exist, it will be created. The file is also the file where **pspell_save_wordlist()** will save personal wordlist to. The file should be writable by whoever PHP runs as (e.g. nobody). Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

Ejemplo 1. pspell_config_personal()

```
<?php
$pspell_config = pspell_config_create("en");
pspell_config_personal($pspell_config, "/var/dictionaries/custom.pws");
$pspell_link = pspell_new_config($pspell_config);
pspell_check($pspell_link, "thecat");
?>
```

pspell_config_repl

(PHP 4 >= 4.0.2, PHP 5)

pspell_config_repl -- Set a file that contains replacement pairs

Description

int **pspell_config_repl** (int dictionary_link, string file)

pspell_config_repl() should be used on a config before calling **pspell_new_config()**. The replacement pairs improve the quality of the spellchecker. When a word is misspelled, and a proper suggestion was not found in the list, **pspell_store_replacement()** can be used to store a replacement pair and then **pspell_save_wordlist()** to save the wordlist along with the replacement pairs. The file should be writable by whoever PHP runs as (e.g. nobody). Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

Ejemplo 1. pspell_config_repl()

```
<?php
$pspell_config = pspell_config_create("en");
pspell_config_personal($pspell_config, "/var/dictionaries/custom.pws");
pspell_config_repl($pspell_config, "/var/dictionaries/custom.repl");
$pspell_link = pspell_new_config($pspell_config);
pspell_check($pspell_link, "thecat");
?>
```

pspell_config_runtogether

(PHP 4 >= 4.0.2, PHP 5)

pspell_config_runtogether -- Consider run-together words as valid compounds

Description

int **pspell_config_runtogether** (int dictionary_link, bool flag)

pspell_config_runtogether() should be used on a config before calling **pspell_new_config()**. This function determines whether run-together words will be treated as legal compounds. That is, "thecat" will be a legal compound, although there should be a space between the two words. Changing this setting only affects the results returned by **pspell_check()**; **pspell_suggest()** will still return suggestions.

Ejemplo 1. pspell_config_runtogether()

```
<?php
$pspell_config = pspell_config_create("en");
pspell_config_runtogether($pspell_config, true);
$pspell_link = pspell_new_config($pspell_config);
pspell_check($pspell_link, "thecat");
?>
```

pspell_config_save_repl

(PHP 4 >= 4.0.2, PHP 5)

pspell_config_save_repl -- Determine whether to save a replacement pairs list along with the wordlist

Description

int **pspell_config_save_repl** (int dictionary_link, bool flag)

pspell_config_save_repl() should be used on a config before calling **pspell_new_config()**. It determines whether **pspell_save_wordlist()** will save the replacement pairs along with the wordlist. Usually there is no need to use this function because if **pspell_config_repl()** is used, the replacement pairs will be saved by **pspell_save_wordlist()** anyway, and if it is not, the replacement pairs will not be saved. Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

pspell_new_config

(PHP 4 >= 4.0.2, PHP 5)

pspell_new_config -- Load a new dictionary with settings based on a given config

Description

int **pspell_new_config** (int config)

pspell_new_config() opens up a new dictionary with settings specified in a config, created with with **pspell_config_create()** and modified with **pspell_config_*()** functions. This method provides you with the most flexibility and has all the functionality provided by **pspell_new()** and **pspell_new_personal()**.

The config parameter is the one returned by **pspell_config_create()** when the config was created.

Ejemplo 1. pspell_new_config()

```
<?php
$pspell_config = pspell_config_create("en");
pspell_config_personal($pspell_config, "/var/dictionaries/custom.pws");
pspell_config_repl($pspell_config, "/var/dictionaries/custom.repl");
$pspell_link = pspell_new_config($pspell_config);
?>
```

pspell_new_personal

(PHP 4 >= 4.0.2, PHP 5)

pspell_new_personal -- Load a new dictionary with personal wordlist

Description

int **pspell_new_personal** (string personal, string language [, string spelling [, string jargon [, string encoding [, int mode]]]])

pspell_new_personal() opens up a new dictionary with a personal wordlist and returns the dictionary link identifier for use in other pspell functions. The wordlist can be modified and saved with **pspell_save_wordlist()**, if desired. However, the replacement pairs are not saved. In order to save replacement pairs, you should create a config using **pspell_config_create()**, set the personal wordlist file with **pspell_config_personal()**, set the file for replacement pairs with **pspell_config_repl()**, and open a new dictionary with **pspell_new_config()**.

The personal parameter specifies the file where words added to the personal list will be stored. It should be an absolute filename beginning with '/' because otherwise it will be relative to $HOME, which is "/root" for most systems, and is probably not what you want.

The language parameter is the language code which consists of the two letter ISO 639 language code and an optional two letter ISO 3166 country code after a dash or underscore.

The spelling parameter is the requested spelling for languages with more than one spelling such as English. Known values are 'american', 'british', and 'canadian'.

The jargon parameter contains extra information to distinguish two different words lists that have the same language and spelling parameters.

The encoding parameter is the encoding that words are expected to be in. Valid values are 'utf-8', 'iso8859-*', 'koi8-r', 'viscii', 'cp1252', 'machine unsigned 16', 'machine unsigned 32'. This parameter is largely untested, so be careful when using.

The mode parameter is the mode in which spellchecker will work. There are several modes available:

- **PSPELL_FAST** - Fast mode (least number of suggestions)
- **PSPELL_NORMAL** - Normal mode (more suggestions)
- **PSPELL_BAD_SPELLERS** - Slow mode (a lot of suggestions)

- **PSPELL_RUN_TOGETHER** - Consider run-together words as legal compounds. That is, "thecat" will be a legal compound, although there should be a space between the two words. Changing this setting only affects the results returned by **pspell_check()**; **pspell_suggest()** will still return suggestions.

Mode is a bitmask constructed from different constants listed above. However, **PSPELL_FAST**, **PSPELL_NORMAL** and **PSPELL_BAD_SPELLERS** are mutually exclusive, so you should select only one of them.

For more information and examples, check out inline manual pspell website:http://aspell.net/.

Ejemplo 1. pspell_new_personal()

```
<?php
$pspell_link = pspell_new_personal ("/var/dictionaries/custom.pws",
        "en", "", "", "", PSPELL_FAST|PSPELL_RUN_TOGETHER);
?>
```

pspell_new

(PHP 4 >= 4.0.2, PHP 5)

pspell_new -- Load a new dictionary

Description

int **pspell_new** (string language [, string spelling [, string jargon [, string encoding [, int mode]]]])

pspell_new() opens up a new dictionary and returns the dictionary link identifier for use in other pspell functions.

The language parameter is the language code which consists of the two letter ISO 639 language code and an optional two letter ISO 3166 country code after a dash or underscore.

The spelling parameter is the requested spelling for languages with more than one spelling such as English. Known values are 'american', 'british', and 'canadian'.

The jargon parameter contains extra information to distinguish two different words lists that have the same language and spelling parameters.

The encoding parameter is the encoding that words are expected to be in. Valid values are 'utf-8', 'iso8859-*', 'koi8-r', 'viscii', 'cp1252', 'machine unsigned 16', 'machine unsigned 32'. This parameter is largely untested, so be careful when using.

The mode parameter is the mode in which spellchecker will work. There are several modes available:

- **PSPELL_FAST** - Fast mode (least number of suggestions)
- **PSPELL_NORMAL** - Normal mode (more suggestions)
- **PSPELL_BAD_SPELLERS** - Slow mode (a lot of suggestions)
- **PSPELL_RUN_TOGETHER** - Consider run-together words as legal compounds. That is,

"thecat" will be a legal compound, although there should be a space between the two words. Changing this setting only affects the results returned by **pspell_check()**; **pspell_suggest()** will still return suggestions.

Mode is a bitmask constructed from different constants listed above. However, **PSPELL_FAST**, **PSPELL_NORMAL** and **PSPELL_BAD_SPELLERS** are mutually exclusive, so you should select only one of them.

For more information and examples, check out inline manual pspell website:http://aspell.net/.

Ejemplo 1. pspell_new()

```
<?php
$pspell_link = pspell_new("en", "", "", "",
                          (PSPELL_FAST|PSPELL_RUN_TOGETHER));
?>
```

pspell_save_wordlist

(PHP 4 >= 4.0.2, PHP 5)

pspell_save_wordlist -- Save the personal wordlist to a file

Description

int **pspell_save_wordlist** (int dictionary_link)

pspell_save_wordlist() saves the personal wordlist from the current session. The dictionary has to be open with **pspell_new_personal()**, and the location of files to be saved specified with **pspell_config_personal()** and (optionally) **pspell_config_repl()**. Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

Ejemplo 1. pspell_add_to_personal()

```
<?php
$pspell_config = pspell_config_create("en");
pspell_config_personal($pspell_config, "/tmp/dicts/newdict");
$pspell_link = pspell_new_config($pspell_config);

pspell_add_to_personal($pspell_link, "Vlad");
pspell_save_wordlist($pspell_link);
?>
```

pspell_store_replacement

(PHP 4 >= 4.0.2, PHP 5)

pspell_store_replacement -- Store a replacement pair for a word

Description

int **pspell_store_replacement** (int dictionary_link, string misspelled, string correct)

pspell_store_replacement() stores a replacement pair for a word, so that replacement can be returned by **pspell_suggest()** later. In order to be able to take advantage of this function, you have to

use **pspell_new_personal()** to open the dictionary. In order to permanently save the replacement pair, you have to use **pspell_config_personal()** and **pspell_config_repl()** to set the path where to save your custom wordlists, and then use **pspell_save_wordlist()** for the changes to be written to disk. Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

Ejemplo 1. pspell_store_replacement()

```
<?php
$pspell_config = pspell_config_create("en");
pspell_config_personal($pspell_config, "/var/dictionaries/custom.pws");
pspell_config_repl($pspell_config, "/var/dictionaries/custom.repl");
$pspell_link = pspell_new_config($pspell_config);

pspell_store_replacement($pspell_link, $misspelled, $correct);
pspell_save_wordlist($pspell_link);
?>
```

pspell_suggest

(PHP 4 >= 4.0.2, PHP 5)

pspell_suggest -- Suggest spellings of a word

Description

array **pspell_suggest** (int dictionary_link, string word)

pspell_suggest() returns an array of possible spellings for the given word.

Ejemplo 1. pspell_suggest() example

```
<?php
$pspell_link = pspell_new("en");

if (!pspell_check($pspell_link, "testt")) {
    $suggestions = pspell_suggest($pspell_link, "testt");

    foreach ($suggestions as $suggestion) {
        echo "Possible spelling: $suggestion<br />";
    }
}
?>
```

CIV. qtdom Functions

Aviso
Esta extensión es *EXPERIMENTAL*. Esto significa que el comportamiento de esta extensión, los nombre de sus funciones y en definitiva TODO lo documentado sobre esta extensión, puede cambiar en una futura versión de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Nota: Esta extensión no está disponible en plataformas Windows

Nota: This extension has been moved to the PECL repository and is no longer bundled with PHP as of PHP 5.0.0.

Requirimientos

You need the Qt-library >=2.2.0

Instalación

Configure PHP --*with-qtdom* to use these functions.

Tabla de contenidos

qdom_error

(PHP 4 >= 4.0.5)

qdom_error -- Returns the error string from the last QDOM operation or FALSE if no errors occurred

Description

string **qdom_error** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

qdom_tree

(PHP 4 >= 4.0.4)

qdom_tree -- Creates a tree of an XML string

Description

QDomDocument **qdom_tree** (string doc)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

CV. Rar Functions

Introducción

Rar is a powerful and effective archiver created by Eugene Roshal. This extension gives you possibility to read Rar archives but doesn't support writing Rar archives, because this is not supported by UnRar library and is directly prohibited by it's license.

More information about Rar and UnRar can be found at http://www.rarlabs.com/.

Requirimientos

No se necesitan bibliotecas externas para construir esta extensión

---

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Instalación

Rar is currently available through PECL http://pecl.php.net/package/rar.

Also you can use the pear installer to install the Rar extension, using the following command: **pear -v install rar**.

You can always download the tar.gz package and install Rar by hand:
Ejemplo 1. Rar installation

```
gunzip rar-xxx.tgz
tar -xvf rar-xxx.tar
cd rar-xxx
phpize
./configure && make && make install
```

Windows users can download the extension dll `php_rar.dll` here:
http://snaps.php.net/win32/PECL_STABLE/.

Tipos de recursos

There is one resource used in Rar extension: a file descriptor returned by **rar_open()**.

Constantes predefinidas

RAR_HOST_MSDOS (integer)

RAR_HOST_OS2 (integer)

RAR_HOST_WIN32 (integer)

RAR_HOST_UNIX (integer)

RAR_HOST_BEOS (integer)

Ejemplos

Ejemplo 2. Rar extension overview example

```
<?php

$rar_file = rar_open('example.rar') or die("Can't open Rar archive");

$entries = rar_list($rar_file);

foreach ($entries as $entry) {
    echo 'Filename: ' . $entry->getName() . "\n";
    echo 'Packed size: ' . $entry->getPackedSize() . "\n";
    echo 'Unpacked size: ' . $entry->getUnpackedSize() . "\n";

    $entry->extract('/dir/extract/to/');
}

rar_close($rar_file);

?>
```

This example opens a Rar file archive and extracts each entry to the specified directory.

Tabla de contenidos

rar_close

(no version information, might be only in CVS)

rar_close -- Close Rar archive and free all resources

Description

bool **rar_close** (resource rar_file)

Close Rar archive and free all allocated resources.

rar_close() returns **FALSE** on error.

rar_entry_get

(no version information, might be only in CVS)

rar_entry_get -- Get entry object from the Rar archive

Description

RarEntry **rar_entry_get** (resource rar_file, string entry_name)

Get entry object from the Rar archive.

Ejemplo 1. Rar::() example

```
<?php

$rar_file = rar_open('example.rar') or die("Failed to open Rar archive");

$entry = rar_entry_get($rar_file, 'Dir/file.txt') or die("Failed to find such entry");

print_r($entry);

?>
```

rar_get_entry() returns entry object or **FALSE** on error.

Rar::extract

(no version information, might be only in CVS)

Rar::extract -- Extract entry from the archive

Description

bool **Rar::extract** (string dir [, string filepath])

Rar::extract() extracts entry's data to the *dir*. It will create new file in the specified *dir* with the name identical to the entry's name. If parameter *filepath* is specified instead *dir*, **Rar::extract()** will extract entry's data to the specified file.

Ejemplo 1. Rar::extract() example

```
<?php

$rar_file = rar_open('example.rar') or die("Failed to open Rar archive");

$entry = rar_entry_get($rar_file, 'Dir/file.txt') or die("Failed to find such entry");

$entry->extract('/dir/to'); // create /dir/to/Dir/file.txt
$entry->extract(false, '/dir/to/new_name.txt'); // create /dir/to/new_name.txt

?>
```

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Rar::getAttr

(no version information, might be only in CVS)

Rar::getAttr -- Get attributes of the entry

Description

int **Rar::getAttr** (void)

Rar::getAttr() returns attributes of the archive entry.

Ejemplo 1. Rar::getAttr() example

```php
<?php

$rar_file = rar_open('example.rar') or die("Can't open Rar archive");

$entry = rar_entry_get($rar_file, 'dir/in/the/archive') or die("Can't find such entry")

$host_os = $entry->getHostOs();
$attr = $entry->getAttr();

switch($host_os) {
    case RAR_HOST_MSDOS:
    case RAR_HOST_OS2:
    case RAR_HOST_WIN32:
    case RAR_HOST_MACOS:
        printf("%c%c%c%c%c%c\n",
                ($attr & 0x08) ? 'V' : '.',
                ($attr & 0x10) ? 'D' : '.',
                ($attr & 0x01) ? 'R' : '.',
                ($attr & 0x02) ? 'H' : '.',
                ($attr & 0x04) ? 'S' : '.',
                ($attr & 0x20) ? 'A' : '.');
        break;
    case RAR_HOST_UNIX:
    case RAR_HOST_BEOS:
        switch ($attr & 0xF000)
        {
            case 0x4000:
                printf("d");
                break;
            case 0xA000:
                printf("l");
                break;
            default:
                printf("-");
                break;
        }
        printf("%c%c%c%c%c%c%c%c%c\n",
                ($attr & 0x0100) ? 'r' : '-',
                ($attr & 0x0080) ? 'w' : '-',
                ($attr & 0x0040) ? (($attr & 0x0800) ? 's':'x'):(($attr & 0x0800) ? 'S'
                ($attr & 0x0020) ? 'r' : '-',
                ($attr & 0x0010) ? 'w' : '-',
                ($attr & 0x0008) ? (($attr & 0x0400) ? 's':'x'):(($attr & 0x0400) ? 'S'
                ($attr & 0x0004) ? 'r' : '-',
                ($attr & 0x0002) ? 'w' : '-',
                ($attr & 0x0001) ? 'x' : '-');
        break;
}

rar_close($rar_file);

?>
```

Rar::getAttr() returns **`FALSE`** on error.

See also **Rar::getHostOs()**.

Rar::getCrc

(no version information, might be only in CVS)

Rar::getCrc -- Get CRC of the entry

Description

int **Rar::getCrc** (void)

Rar::getCrc() returns CRC of the archive entry.

Ejemplo 1. Rar::getCrc() example

```
<?php

?>
```

Rar::getCrc() returns **FALSE** on error.

Rar::getFileTime

(no version information, might be only in CVS)

Rar::getFileTime -- Get entry last modification time

Description

string **Rar::getFileTime** (void)

Rar::getFileTime() returns entry last modification time as string in format YYYY-MM-DD HH:II:SS.

Ejemplo 1. Rar::() example

```
<?php

?>
```

Rar::getFileTime() returns **FALSE** on error.

Rar::getHostOs

(no version information, might be only in CVS)

Rar::getHostOs -- Get entry host OS

Description

int **Rar::getHostOs** (void)

Rar::getHostOs() return code of the host OS of the archive entry.

Ejemplo 1. Rar::getHostOs() example

```
<?php

$rar_file = rar_open('example.rar') or die("Failed to open Rar archive");

$entry = rar_entry_get($rar_file, 'Dir/file.txt') or die("Failed to find such entry");

switch ($entry->getHostOs()) {
    case RAR_HOST_MSDOS:
        echo "MS-DOS\n";
        break;
    case RAR_HOST_OS2:
        echo "OS2\n";
        break;
    case RAR_HOST_WIN32:
        echo "Win32\n";
        break;
    case RAR_HOST_MACOS:
        echo "MacOS\n";
        break;
    case RAR_HOST_UNIX:
        echo "Unix/Linux\n";
        break;
    case RAR_HOST_BEOS:
        echo "BeOS\n";
        break;
}

?>
```

Rar::getHostOs() returns **FALSE** on error.

Rar::getMethod

(no version information, might be only in CVS)

Rar::getMethod -- Get pack method of the entry

Description

int **Rar::getMethod** (void)

Rar::getMethod() returns number of the method used when adding current archive entry.

Ejemplo 1. Rar::getMethod() example

```
<?php

$rar_file = rar_open('example.rar') or die("Failed to open Rar archive");

$entry = rar_entry_get($rar_file, 'Dir/file.txt') or die("Failed to find such entry");

echo "Method number: " . $entry->getMethod();

?>
```

Rar::getMethod() returns **FALSE** on error.

Rar::getName

(no version information, might be only in CVS)

Rar::getName -- Get name of the entry

Description

string **Rar::getName** (void)

Rar::getName() returns full name of the archive entry.

Ejemplo 1. Rar::getName() example

```
<?php
$rar_file = rar_open('example.rar') or die("Failed to open Rar archive");
$entry = rar_entry_get($rar_file, 'Dir/file.txt') or die("Failed to find such entry");
echo "Entry name: " . $entry->getName();
?>
```

Rar::getName() returns **FALSE** on error.

Rar::getPackedSize

(no version information, might be only in CVS)

Rar::getPackedSize -- Get packed size of the entry

Description

int **Rar::getPackedSize** (void)

Get packed size of the archive entry.

Ejemplo 1. Rar::() example

```
<?php
$rar_file = rar_open('example.rar') or die("Failed to open Rar archive");
$entry = rar_entry_get($rar_file, 'Dir/file.txt') or die("Failed to find such entry");
echo "Packed size of " . $entry->getName() . " = " . $entry->getPackedSize() . " bytes"
?>
```

Rar::getPackedSize() returns **FALSE** on error.

Rar::getUnpackedSize

(no version information, might be only in CVS)

Rar::getUnpackedSize -- Get unpacked size of the entry

Description

int **Rar::getUnpackedSize** (void)

Get unpacked size of the archive entry.

Ejemplo 1. Rar::getUnpackedSize() example

```
<?php
$rar_file = rar_open('example.rar') or die("Failed to open Rar archive");
$entry = rar_entry_get($rar_file, 'Dir/file.txt') or die("Failed to find such entry");
echo "Unpacked size of " . $entry->getName() . " = " . $entry->getPackedSize() . " byte
?>
```

Rar::getUnpackedSize() returns **FALSE** on error.

Rar::getVersion

(no version information, might be only in CVS)

Rar::getVersion -- Get version of the archiver used to add the entry

Description

int **Rar::getVersion** (void)

Get version of the archiver used to add the archive entry.

Ejemplo 1. Rar::getVersion() example

```
<?php
$rar_file = rar_open('example.rar') or die("Failed to open Rar archive");
$entry = rar_entry_get($rar_file, 'Dir/file.txt') or die("Failed to find such entry");
echo "Rar (WinRAR) version used: " . $entry->getVersion();
?>
```

Rar::getVersion() returns **FALSE** on error.

rar_list

(no version information, might be only in CVS)

rar_list -- Get entries list from the Rar archive

Description

array **rar_list** (resource rar_file)

Get entries list from the Rar archive.

Ejemplo 1. Rar::() example

```
<?php

$rar_file = rar_open('example.rar') or die("Failed to open Rar archive");

$entries_list = rar_list($rar_file);

print_r($entries_list);

?>
```

rar_list() returns array of entries or **FALSE** on error.

rar_open

(no version information, might be only in CVS)

rar_open -- Open Rar archive

Description

resource **rar_open** (string filename [, string password])

Open specified Rar archive and return Rar file resource.

rar_open() returns Rar file resource or **FALSE** on error.

CVI. GNU Readline

Introducción

Las funciones **readline()** implementan una interfaz con la biblioteca GNU Readline. Estas son funciones que ofrecen líneas de comando editables. Un ejemplo de la manera en que trabajan podría ser la forma en que Bash le permite usar las teclas de flechas para insertar caracteres o desplazarse a través del historial de comandos. Debido a la naturaleza interactiva de esta biblioteca, tendrá un uso muy reducido en la escritura de aplicaciones Web, aunque puede ser útil cuando se escriben scripts al usar PHP desde la línea de comandos.

> **Nota:** Esta extensión no está disponible en plataformas Windows

---

Requirimientos

Para usar las funciones readline, necesita instalar libreadline. Puede encontrar libreadlinea en la página web del proyecto GNU Readline, en http://cnswww.cns.cwru.edu/~chet/readline/rltop.html. Este proyecto es administrado por Chet Ramey, quien es también el autor de Bash.

También puede usar estas funciones con la biblioteca libedit, un reemplazo no-GPL de la biblioteca readline. La biblioteca libedit es distribuida bajo una licencia BSD y está disponible para su descarga en http://sourceforge.net/projects/libedit/.

Instalación

Para usar estas funciones, debe compilar la versión CGI o CLI de PHP con soporte para readline. Necesita configurar PHP con la opción *--with-readline[=DIR]*. Si desea usar el reemplazo de readline, libedit, configure PHP con la opción *--with-libedit[=DIR]*.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Esta extensión no tiene ninguna constante definida.

Tabla de contenidos

readline_add_history

(PHP 4 , PHP 5)

readline_add_history -- Añade una línea al historial

Descripción

void **readline_add_history** (string line)

Esta función añade una línea al historial de líneas de comandos.

readline_callback_handler_install

(no version information, might be only in CVS)

readline_callback_handler_install -- Initializes the readline callback interface and terminal, prints the prompt and returns immediately

Descripción

bool **readline_callback_handler_install** (string prompt, callback callback)

Sets up a readline callback interface then prints *prompt* and immediately returns. The *callback* function takes one parameter; the user input returned. Calling this function twice without removing the previous callback interface will automatically and conveniently overwrite the old interface.

The callback feature is useful when combined with **stream_select()** as it allows interleaving of IO and user input, unlike **readline()**.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplos

Ejemplo 1. Readline Callback Interface Example

```
<?php
function rl_callback($ret)
{
    global $c, $prompting;

    echo "You entered: $ret\n";
    $c++;

    if ($c > 10) {
        $prompting = false;
        readline_callback_handler_remove();
    } else {
        readline_callback_handler_install("[$c] Enter something: ", 'rl_callback');
    }
}

$c = 1;
$prompting = true;

readline_callback_handler_install("[$c] Enter something: ", 'rl_callback');

while ($prompting) {
    $n = stream_select($r = array(STDIN), $w = null, $e = null, null);
    if ($n && in_array(STDIN, $r)) {
        // read a character, will call the callback when a newline is entered
        readline_callback_read_char();
    }
}

echo "Prompting disabled. All done.\n";
?>
```

Ver también

readline_callback_handler_remove(), readline_callback_read_char(), y stream_select().

readline_callback_handler_remove

(no version information, might be only in CVS)

readline_callback_handler_remove -- Removes a previously installed callback handler and restores terminal settings

Descripción

bool **readline_callback_handler_remove** (void)

Removes a previously installed callback handler and restores terminal settings.

Ejemplos

See readline_callback_handler_install() for an example of how to use the readline callback interface.

Valores retornados

Returns **TRUE** if a previously installed callback handler was removed, or **FALSE** if one could not be found.

Ver también

readline_callback_handler_install(), and readline_callback_read_char().

readline_callback_read_char

(no version information, might be only in CVS)

readline_callback_read_char -- Reads a character and informs the readline callback interface when a line is received

Descripción

void **readline_callback_read_char** (void)

Reads a character of user input. When a line is received, this function informs the readline callback interface installed using readline_callback_handler_install() that a line is ready for input.

Ejemplos

See readline_callback_handler_install() for an example of how to use the readline callback interface.

Ver también

readline_callback_handler_install(), and **readline_callback_handler_remove()**.

readline_clear_history

(PHP 4 , PHP 5)

readline_clear_history -- Borra el historial

Descripción

boolean **readline_clear_history** (void)

Esta función borra por completo el historial de la línea de comandos.

readline_completion_function

(PHP 4 , PHP 5)

readline_completion_function -- Registra una función de completitud

Descripción

boolean **readline_completion_function** (string line)

Esta función registra una función de completitud. Debe proporcionar el nombre de una función existente que acepte una línea de comandos parcial y devuelva una array con posibles coincidencias. Es el mismo tipo de funcionalidad que se obtiene al pulsar la tecla de tabulación cuando se está usando el Bash.

readline_info

(PHP 4 , PHP 5)

readline_info -- Establece/Obtiene diversas variables internas de readline

Descripción

mixed **readline_info** ([string varname [, string newvalue]])

Si es llamada sin parámetros, esta función devuelve un array con los valores de todas las opciones que readline usa. Los elementos vendrán indexados por los siguientes valores: done, end, erase_empty_line, library_version, line_buffer, mark, pending_input, point, prompt, readline_name, y terminal_name.

Si es llamada con un parámetros, devuelve el valor de esa opción. Si es llamada con dos parámetros, el valor de la opción será cambiado al parámetro dado.

readline_list_history

(PHP 4 , PHP 5)

readline_list_history -- Lista el historial

Descripción

array **readline_list_history** (void)

Esta función devuelve un array con el historial de líneas de comandos completo. Los elementos están indexados por enteros comenzando por el cero.

readline_on_new_line

(no version information, might be only in CVS)

readline_on_new_line -- Inform readline that the cursor has moved to a new line

Description

void **readline_on_new_line** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

readline_read_history

(PHP 4 , PHP 5)

readline_read_history -- Lee el historial

Descripción

bool **readline_read_history** ([string nombre_archivo])

Esta función lee un historial de comandos desde un archivo.

readline_redisplay

(no version information, might be only in CVS)

readline_redisplay -- Ask readline to redraw the display

Description

void **readline_redisplay** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

readline_write_history

(PHP 4 , PHP 5)

readline_write_history -- Escribe el historial

Descripción

boolean **readline_write_history** (string filename)

Esta función escribe el historial de comandos en un archivo.

readline

(PHP 4 , PHP 5)

readline -- Lee una línea

Descripción

string **readline** ([string prompt])

Esta función devuelve una única cadena del usuario. Puede especificar una cadena que se mostrará al usuario. La línea devuelta tiene el indicador final de nueva línea eliminado. Necesita añadir esta línea al historial usando la función **readline_add_history()**.

Ejemplo 1. readline()

```
<?php
//obtiene 3 comandos del usuario
for ($i=0; $i < 3; $i++) {
        $line = readline ("Comando: ");
        readline_add_history ($line);
}

//Vuelca el historial
print_r (readline_list_history());

//Vuelca las variables
print_r (readline_info());
?>
```

CVII. Funciones GNU Recode

Introducción

Este módulo contiene una interfaz con la biblioteca GNU Recode. La biblioteca GNU Recode convierte archivos entre varios juegos de caracteres codificados y codificaciones de superficie. Cuando esto no puede conseguirse con exactitud, puede que se deshaga de los caracteres problemáticos o recurra a aproximaciones. La biblioteca reconoce o produce cerca de 150 juegos de caracteres diferentes y es capaz de convertir archivos entre casi cualquier par de ellos. La mayoría de juegos de caracteres del documento RFC 1345 se encuentran soportados.

> **Nota:** Esta extensión no está disponible en plataformas Windows

---

Requirimientos

Usted debe tener instalado GNU Recode 3.5 o superior en su sistema. Puede descargar el paquete desde http://www.gnu.org/directory/All_GNU_Packages/recode.html.

Aviso
La versión 3.6 de la biblioteca Recode agrega caracteres extraños detrás de cadenas convertidas bajo ciertas circunstancias. Por lo tanto es más seguro usar Recode v3.5 o alguna de las alternativas disponibles, como las extensiones iconv o mbstring.

Instalación

To be able to use the functions defined in this module you must compile your PHP interpreter using the *--with-recode[=DIR]* option.

Aviso
Crashes and startup problems of PHP may be encountered when loading the recode as extension **after** loading any extension of mysql or imap. Loading the recode before those extension has proved to fix the problem. This is due a technical problem that both the c-client library used by imap and recode have their own *hash_lookup()* function and both mysql and recode have their own *hash_insert* function.

Aviso
La extensión IMAP no puede ser usada junto con las extensiones recode, YAZ ó Cyrus. Esto es debido a que las dos utilizan el mismo símbolo interno

---

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Esta extensión no tiene ninguna constante definida.

Tabla de contenidos

recode_file

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

recode_file -- Recodificar de un archivo a otro de acuerdo a la petición de recodificación

Descripción

bool **recode_file** (string peticion, resource entrada, resource salida)

Recodificar el archivo referenciado por el gestor de archivo *entrada* en el archivo referenciado por el gestor de archivo *salida* de acuerdo a la *peticion* de recodificación. Devuelve **FALSE** si no es posible cumplir la petición, o **TRUE** de lo contrario.

En la actualidad, esta función no procesa gestores de archivo que hagan referencia a archivos remotos (URLs). Ambos gestores de archivo deben hacer referencia a archivos locales.

Ejemplo 1. Ejemplo básico de recode_file()

```
<?php
$entrada = fopen('entrada.txt', 'r');
$salida = fopen('salida.txt', 'w');
recode_file("us..flat", $entrada, $salida);
?>
```

recode_string

(PHP 3>= 3.0.13, PHP 4 , PHP 5)

recode_string -- Recodifica una cadena literal segun una peticion de recodificacion.

Description

string **recode_string** (string request, string string)

Recodifica la cadena *string* segun una peticion de recodificacion *request*. Devuelve **FALSE** si no puede realizar la recodificacion, **TRUE** si todo va bien.

Una simple peticion "recode" podria ser "lat1..iso646-de". Ver tambien la documentacion de GNU Recode de tu instalacion para obtener instrucciones detalladas sobre peticiones de recodificacion.

recode

recode -- Alias of **recode_string()**

Description

This function is an alias of **recode_string()**.

CVIII. Funciones de Expresiones Regulares (POSIX Extendido)

Introducción

> **Sugerencia:** PHP ofrece también soporte de expresiones regulares usando una sintaxis compatible con Perl usando las Funciones PCRE. Tales funciones soportan coincidencias no-ambiciosas, aserciones, subpatrones condicionales, y un número de características adicionales que no son soportadas por las sintaxis de expresiones regulares POSIX-extendido.

Aviso
Estas expresiones regulares no son seguras con material binario. Las Funciones PCRE lo son.

Las expresiones regulares son usadas para la manipulación compleja de cadenas. PHP usa expresiones regulares POSIX extendidas, tal y como se definen por POSIX 1003.2. Para una descripción completa de las expresiones regulares POSIX, vea las páginas de manual de regex incluidas en el directorio regex en la distribución de PHP. Se encuentran en formato manpage, así que querrá hacer algo del estilo de **man /usr/local/src/regex/regex.7** para leerlas.

Requirimientos

No se necesitan bibliotecas externas para construir esta extensión

---

Instalación

Aviso
No modifique el TIPO a menos que sepa lo que está haciendo.

Para habilitar el soporte para expresiones regulares, configure PHP con la opción *--with-regex [=TIPO]*. TIPO puede ser un valor entre system, apache, php. La acción predeterminada es usar php.

La versión para Windows de *PHP* tiene soporte nativo para esta extensión. No se necesita cargar ninguna extensión adicional para usar estas funciones.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Esta extensión no tiene ninguna constante definida.

Ejemplos

Ejemplo 1. Ejemplos de Expresiones Regulares

```
<?php
// Devuelve true si "abc" se encuentra en cualquier lugar de $cadena.
ereg("abc", $cadena);

// Devuelve true si "abc" es encontrado al comienzo de $cadena
ereg("^abc", $cadena);

// Devuelve true si "abc" es encontrado al final de $cadena.
ereg("abc$", $cadena);

// Devuelve true si el navegador del cliente es Netscape 2, 3 or MSIE 3.
eregi("(ozilla.[23]|MSIE.3)", $HTTP_USER_AGENT);

// Coloca tres palabras separadas por espacios en $regs[1], $regs[2] y $regs[3].
ereg("([[:alnum:]]+) ([[:alnum:]]+) ([[:alnum:]]+)", $cadena, $regs);

// Coloca una etiqueta <br /> al comienzo de $cadena.
$cadena = ereg_replace("^", "<br />", $cadena);

// Coloca una etiqueta <br /> al final de $cadena.
$cadena = ereg_replace("$", "<br />", $cadena);

// Se deshace de cualquier caracter de salto de linea en $cadena.
$cadena = ereg_replace("\n", "", $cadena);
?>
```

Ver también

Para consultar sobre expresiones regulares en una sintaxis compatible con Perl, eche un vistazo a las Funciones PCRE. Las coincidencias más simples de comodines tipo intérprete de comandos son ofrecidas por **fnmatch()**.

Tabla de contenidos

eregi_replace -- reemplaza expresiones regularse sin diferenciar mayúsculas y minúsculas
eregi -- coincidencia de expresiones regulares sin diferenciar mayúsculas y minúsculas
split -- divide la cadena en elementos de un array según una expresión regular
spliti -- Separar una cadena en una matriz mediante una expresión regular, no sensible a mayúsculas ni minúsculas
sql_regcase -- construye una expresión regular para buscar coincidencias sin diferenciar mayúsculas y minúsculas

ereg_replace

(PHP 3, PHP 4 , PHP 5)

ereg_replace -- reemplaza expresiones regulares

Descripción

string **ereg_replace** (string pattern, string replacement, string string)

Esta función examina *string* buscando coincidencias de *pattern*, y reemplaza el texto encontrado con *replacement*.

Devuelve la cadena modificada. Si no hay coincidencias que reemplazar, devuelve la cadena original.

Si *pattern* contiene subcadenas entre paréntesis, *replacement* puede contener subcadenas de la forma *cifra*, que serán reemplazadas por el texto que coincide con la subcadena entre paréntesis que ocupa el lugar indicado por la cifra; *0* produce el contenido total de la cadena. Se pueden usar hasta nueve subcadenas. Los paréntesis pueden anidarse; en este caso se cuentan los paréntesis de apertura.

Si no se encuentran coincidencias en *string*, se devuelve *string* sin cambios.

Por ejemplo, el siguiente fragmento de código imprime "This was a test" tres veces:
Ejemplo 1. ereg_replace() example

```
$string = "This is a test";
echo ereg_replace( " is", " was", $string );
echo ereg_replace( "( )is", "\\1was", $string );
echo ereg_replace( "(( )is)", "\\2was", $string );
```

Ver también **ereg()**, **eregi()**, y **eregi_replace()**.

ereg

(PHP 3, PHP 4 , PHP 5)

ereg -- Coincidencia de expresiones regulares

Descripción

int **ereg** (string pattern, string string [, array regs])

Busca en *string* las coincidencias con la expresión regular *pattern*.

Si se encuentran coincidencias con subcadenas entre paréntesis de *pattern* y la función se ha llamado con el tercer argumento *regs*, las coincidencias se almacenarán en los elementos de *regs*. $regs[1] contendrá la subcadena que empieza en el primer paréntesis izquierdo; $regs[2] la que comienza en el segundo, etc. $regs[0] contendrá una copia de *string*.

La búsqueda diferencia mayúsculas y minúsculas.

Devuelve un valor verdadero si se encontró alguna coincidencia, o falso in no se encontraron coincidencias u ocurrió algún error.

El siguiente fragmento de código toma una fecha en formato ISO (AAAA-MM-DD) y la imprime en formato DD.MM.AAAA:

Ejemplo 1. ereg() example

```
if ( ereg( "([0-9]{4})-([0-9]{1,2})-([0-9]{1,2})", $date, $regs ) ) {
    echo "$regs[3].$regs[2].$regs[1]";
} else {
    echo "Invalid date format: $date";
}
```

Ver también **eregi()**, **ereg_replace()**, y **eregi_replace()**.

eregi_replace

(PHP 3, PHP 4 , PHP 5)

eregi_replace -- reemplaza expresiones regularse sin diferenciar mayúsculas y minúsculas

Descripción

string **eregi_replace** (string pattern, string replacement, string string)

Esta función es idéntica a **ereg_replace()**, excepto en que ignora la distinción entre mayúsculas y minúsculas.

Ver también **ereg()**, **eregi()**, y **ereg_replace()**.

eregi

(PHP 3, PHP 4 , PHP 5)

eregi -- coincidencia de expresiones regulares sin diferenciar mayúsculas y minúsculas

Descripción

int **eregi** (string pattern, string string [, array regs])

Esta función es idéntica a **ereg()**, excepto en que ignora la distinción entre mayúsculas y minúsculas.

Ver también **ereg()**, **ereg_replace()**, y **eregi_replace()**.

split

(PHP 3, PHP 4 , PHP 5)

split -- divide la cadena en elementos de un array según una expresión regular

Descripción

array **split** (string pattern, string string [, int limit])

Devuelve un array de cadenas, cada una de las cuales es una subcadena de *string* formada al dividir esta en los límites formados por la expresión regular *pattern*. Si ocurre un error, devuelve un valor falso.

Para obtener los cinco primeros campos de una línea de `/etc/passwd`:
Ejemplo 1. split() example

```
$passwd_list = split( ":", $passwd_line, 5 );
```

Para examinar una fecha que puede estar delimitada por barras, puntos o guiones:
Ejemplo 2. split() example

```
$date = "04/30/1973";  // Los delimitadores pueden ser barras, puntos o guiones
list( $month, $day, $year ) = split( '[/.-]', $date );
echo "Month: $month; Day: $day; Year: $year<br>\n";
```

Observar que *pattern* distingue entre mayúsculas y minúsculas.

Observar que si no se necesita la potencia de las expresiones regulares, es más rápido utilizar **explode()**, que no carga el motor de expresiones regulares.

Por favor, observar que *pattern* es una expresión regular. Si se quiere dividir con alguno de los caracteres especiales de las expresiones regulares, se necesita protegerlo antes. Si pareciera que **split ()** (o cualquier otra función de regex) está haciendo algo irregular, léase el archivo `regex.7`, incluido en el subdirectorio `regex` de la distribución de PHP. Está en formato de página de manual, por lo que para leerlo es necesaria una orden como **man /usr/local/src/regex/regex.7**.

Ver también: **explode()** e **implode()**.

spliti

(PHP 4 >= 4.0.1, PHP 5)

spliti -- Separar una cadena en una matriz mediante una expresión regular, no sensible a mayúsculas ni minúsculas

Descripción

array **spliti** (string patron, string cadena [, int limite])

Esta función es idéntica a **split()**, excepto que ignora la distinción entre mayúsculas y minúsculas cuando realiza coincidencias sobre caracteres alfabéticos.

Vea también **preg_split()**, **split()**, **explode()**, e **implode()**.

sql_regcase

(PHP 3, PHP 4 , PHP 5)

sql_regcase -- construye una expresión regular para buscar coincidencias sin diferenciar mayúsculas y minúsculas

Descripción

string **sql_regcase** (string string)

Devuelve una expresión regular válida que coincide con *string* sin distinguir mayúsculas y minúsculas. Esta expresión es *string* con cada carácter convertido a una expresión entre corchetes que contiene el carácter en mayúscula y minúscula, si es posible; en caso contrario, contiene el carácter original dos veces.

Ejemplo 1. sql_regcase() example

```
echo sql_regcase( "Foo bar" );
```

imprime

```
[Ff][Oo][Oo][  ][Bb][Aa][Rr]
```

.

Se puede utilizar para lograr coincidencias que no diferencien mayúsculas de minúsculas en productos que sólo soportan expresiones regulares que sí distinguen.

CIX. Funciones Semáforo y de memoria compartida

Este módulo provee funciones semáforo utilizando los semaforos de System V. Los semáforos pueden usarse para obtener acceso exclusivo a algun recurso del ordenador en cuestión, o para limitar el número de procesos que pueden usar un recurso simultaneamente.

Este módulo provee tambien funciones de memoria compartida, usando el compartimiento de memoria de System V. La memoria compartida puede usarse para proveer acceso a variables globales. Los diferentes demonios http e incluso otros programas, (como Perl, C, ...) son capaces de utilizar estos datos, para intercambiarlos de modo global. Recuerde que, la memoria compartida NO es segura para los accesos simultáneos. Use los semáforos para obtener sincronismo.

Tabla 1. Limites de la memoria compartida del SO Unix

SHMMAX	máximo tamaño de memoria compartida, normalmente 131072 bytes
SHMMIN	minimo tamaño de memoria compartida, por lo general 1 byte
SHMMNI	máxima cantidad de segmentos de memoria compartida, normalmente 100
SHMSEG	máximo de memoria compartida por proceso, normalmente 6

Tabla de contenidos

msg_remove_queue -- Destroy a message queue
msg_send -- Send a message to a message queue
msg_set_queue -- Set information in the message queue data structure
msg_stat_queue -- Returns information from the message queue data structure
sem_acquire -- adquiere un semáforo, lo toma para sí
sem_get -- obtiene la identificacion de un semáforo (semaphore id)
sem_release -- release a semaphore
sem_remove -- Remove a semaphore
shm_attach -- Crea o abre un segmento de memoria compartida
shm_detach -- Finaliza conexión con un segmento de memoria compartida
shm_get_var -- Devuelve una variable de la memoria compartida
shm_put_var -- Inserta o actualiza una variable en la memoria compartida
shm_remove_var -- Elimina una variable de la memoria compartida
shm_remove -- Elimina memoria compartida del sistma Unix

ftok

(PHP 4 >= 4.2.0, PHP 5)

ftok -- Convert a pathname and a project identifier to a System V IPC key

Description

int **ftok** (string pathname, string proj)

The function converts the *pathname* of an existing accessible file and a project identifier (*proj*) into a *integer* for use with for example **shmop_open()** and other System V IPC keys. The *proj* parameter should be a one character string.

On success the return value will be the created key value, otherwise *-1* is returned.

See also **shmop_open()** and **sem_get()**.

msg_get_queue

(PHP 4 >= 4.3.0, PHP 5)

msg_get_queue -- Create or attach to a message queue

Description

resource **msg_get_queue** (int key [, int perms])

msg_get_queue() returns an id that can be used to access the System V message queue with the given *key*. The first call creates the message queue with the optional *perms* (default: 0666). A second call to **msg_get_queue()** for the same *key* will return a different message queue identifier, but both identifiers access the same underlying message queue. If the message queue already exists, the *perms* will be ignored.

See also **msg_remove_queue()**, **msg_receive()**, **msg_send()**, **msg_stat_queue()** and **msg_set_queue()**.

msg_receive

(PHP 4 >= 4.3.0, PHP 5)

msg_receive -- Receive a message from a message queue

Description

bool **msg_receive** (resource queue, int desiredmsgtype, int &msgtype, int maxsize, mixed &message [, bool unserialize [, int flags [, int &errorcode]]])

msg_receive() will receive the first message from the specified *queue* of the type specified by *desiredmsgtype*. The type of the message that was received will be stored in *msgtype*. The maximum size of message to be accepted is specified by the *maxsize*; if the message in the queue is larger than this size the function will fail (unless you set *flags* as described below). The received message will be stored in *message*, unless there were errors receiving the message, in which case the optional *errorcode* will be set to the value of the system errno variable to help you identify the cause.

If *desiredmsgtype* is 0, the message from the front of the queue is returned. If *desiredmsgtype* is greater than 0, then the first message of that type is returned. If *desiredmsgtype* is less than 0, the first message on the queue with the lowest type less than or equal to the absolute value of *desiredmsgtype* will be read. If no messages match the criteria, your script will wait until a suitable message arrives on the queue. You can prevent the script from blocking by specifying **MSG_IPC_NOWAIT** in the *flags* parameter.

unserialize defaults to **TRUE**; if it is set to **TRUE**, the message is treated as though it was serialized using the same mechanism as the session module. The message will be unserialized and then returned to your script. This allows you to easily receive arrays or complex object structures from other PHP scripts, or if you are using the WDDX serializer, from any WDDX compatible source. If *unserialize* is **FALSE**, the message will be returned as a binary-safe string.

The optional *flags* allows you to pass flags to the low-level msgrcv system call. It defaults to 0, but you may specify one or more of the following values (by adding or ORing them together).

Tabla 1. Flag values for msg_receive

MSG_IPC_NOWAIT	If there are no messages of the *desiredmsgtype*, return immediately and do not wait. The function will fail and return an integer value corresponding to ENOMSG.
MSG_EXCEPT	Using this flag in combination with a *desiredmsgtype* greater than 0 will cause the function to receive the first message that is not equal to *desiredmsgtype*.
MSG_NOERROR	If the message is longer than *maxsize*, setting this flag will truncate the message to *maxsize* and will not signal an error.

Upon successful completion the message queue data structure is updated as follows: *msg_lrpid* is set to the process-ID of the calling process, *msg_qnum* is decremented by 1 and *msg_rtime* is set to the current time.

msg_receive() returns **TRUE** on success or **FALSE** on failure. If the function fails, the optional *errorcode* will be set to the value of the system errno variable.

See also **msg_remove_queue()**, **msg_send()**, **msg_stat_queue()** and **msg_set_queue()**.

msg_remove_queue

(PHP 4 >= 4.3.0, PHP 5)

msg_remove_queue -- Destroy a message queue

Description

bool **msg_remove_queue** (resource queue)

msg_remove_queue() destroys the message queue specified by the *queue*. Only use this function when all processes have finished working with the message queue and you need to release the system resources held by it.

See also **msg_remove_queue()**, **msg_receive()**, **msg_stat_queue()** and **msg_set_queue()**.

msg_send

(PHP 4 >= 4.3.0, PHP 5)

msg_send -- Send a message to a message queue

Description

bool **msg_send** (resource queue, int msgtype, mixed message [, bool serialize [, bool blocking [, int &errorcode]]])

msg_send() sends a *message* of type *msgtype* (which MUST be greater than 0) to a the message queue specified by *queue*.

If the message is too large to fit in the queue, your script will wait until another process reads messages from the queue and frees enough space for your message to be sent. This is called blocking; you can prevent blocking by setting the optional *blocking* parameter to **FALSE**, in which case **msg_send()** will immediately return **FALSE** if the message is too big for the queue, and set the optional *errorcode* to EAGAIN, indicating that you should try to send your message again a little later on.

The optional *serialize* controls how the *message* is sent. *serialize* defaults to **TRUE** which means that the *message* is serialized using the same mechanism as the session module before being sent to the queue. This allows complex arrays and objects to be sent to other PHP scripts, or if you are using the WDDX serializer, to any WDDX compatible client.

Upon successful completion the message queue data structure is updated as follows: *msg_lspid* is set to the process-ID of the calling process, *msg_qnum* is incremented by 1 and *msg_stime* is set to the current time.

See also **msg_remove_queue()**, **msg_receive()**, **msg_stat_queue()** and **msg_set_queue()**.

msg_set_queue

(PHP 4 >= 4.3.0, PHP 5)

msg_set_queue -- Set information in the message queue data structure

Description

bool **msg_set_queue** (resource queue, array data)

msg_set_queue() allows you to change the values of the msg_perm.uid, msg_perm.gid, msg_perm.mode and msg_qbytes fields of the underlying message queue data structure. You specify the values you require by setting the value of the keys that you require in the *data* array.

Changing the data structure will require that PHP be running as the same user that created the queue, owns the queue (as determined by the existing msg_perm.xxx fields), or be running with root privileges. root privileges are required to raise the msg_qbytes values above the system defined limit.

See also **msg_remove_queue()**, **msg_receive()**, **msg_stat_queue()** and **msg_set_queue()**.

msg_stat_queue

(PHP 4 >= 4.3.0, PHP 5)

msg_stat_queue -- Returns information from the message queue data structure

Description

array **msg_stat_queue** (resource queue)

msg_stat_queue() returns the message queue meta data for the message queue specified by the *queue*. This is useful, for example, to determine which process sent the message that was just received.

The return value is an array whose keys and values have the following meanings:

Tabla 1. Array structure for msg_stat_queue

msg_perm.uid	The uid of the owner of the queue.
msg_perm.gid	The gid of the owner of the queue.
msg_perm.mode	The file access mode of the queue.
msg_stime	The time that the last message was sent to the queue.
msg_rtime	The time that the last message was received from the queue.
msg_ctime	The time that the queue was last changed.
msg_qnum	The number of messages waiting to be read from the queue.

msg_qbytes	The number of bytes of space currently available in the queue to hold sent messages until they are received.
msg_lspid	The pid of the process that sent the last message to the queue.
msg_lrpid	The pid of the process that received the last message from the queue.

See also **msg_remove_queue()**, **msg_receive()**, **msg_stat_queue()** and **msg_set_queue()**.

sem_acquire

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

sem_acquire -- adquiere un semáforo, lo toma para sí

Descripción

int **sem_acquire** (int sem_identifier)

Devuelve: Verdadero si hay éxito, falso si hay errores

sem_acquire() Produce un bloqueo (de ser necesario) hasta que el semáforo puede adquirirse. Un proceso intentando adquirir un semáforo ya adquirido, se bloqueará permanentemente si adquiriendo el semáforo, excede su valor de max_acquire.

Despues de procesar una petición, cualquier semáforo adquirido por un proceso, que no sea explícitamente liberado, será liberado automñaticamente, generando un mensaje de alerta.

Véase tambien: **sem_get()** and **sem_release()**.

sem_get

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

sem_get -- obtiene la identificacion de un semáforo (semaphore id)

Descripción

int **sem_get** (int key [, int max_acquire [, int perm]])

Devuelve: Un identificador positivo de semáforo en caso de éxito, o falso en caso de error.

sem_get() Devuelve un id que puede usarse para acceder al semáforo de System V con la llave dada. El semáforo se usa si es necesario usando los bits de permisos especificados en perm (por defecto 0666). El número de procesos que pueden adquirir el semáforo simultáneamente, se define en max_acquire (por defecto es 1). Actualmente este valor se fija sólo si el proceso determina que es el único relacionado actualmente al semáforo.

Una segunda llamada a **sem_get()** para la misma llave, devolverá un id de semáforo diferente, pero con ambos identificadoes, se accederá al mismo semáforo.

Véase también: **sem_acquire()** y **sem_release()**.

sem_release

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

sem_release -- release a semaphore

Descripción

int **sem_release** (int sem_identifier)

Returns: **TRUE** on success, **FALSE** on error

sem_release() releases the semaphore if it is currently acquired by the calling process, otherwise a warning is generated.

After releasing the semaphore, **sem_acquire()** may be called to re-acquire it.

Véase tambien: **sem_get()** y **sem_acquire()**.

sem_remove

(PHP 4 >= 4.1.0, PHP 5)

sem_remove -- Remove a semaphore

Description

bool **sem_remove** (resource sem_identifier)

sem_remove() removes the semaphore *sem_identifier* if it has been created by **sem_get()**, otherwise generates a warning.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

After removing the semaphore, it is no more accessible.

See also **sem_get()**, **sem_release()** and **sem_acquire()**.

shm_attach

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

shm_attach -- Crea o abre un segmento de memoria compartida

Descripción

int **shm_attach** (int key [, int memsize [, int perm]])

shm_attach() devuelve un identificador que puede usarse para acceder a la memoria compartida de

SystemV, con la llave dada, la primera llamada creará el segmento de memoria compartida con mem_size de tamaño (por defecto: sysvshm.init_mem en el archivo de configuración, o bien de 10000 bytes) y los bits de permiso mas apropiados (por defecto: 0666).

Una segunda llamada a **shm_attach()** con la misma *llave* devolvera un id diferente de memoria compartida, pero ambos identificadores acceden a la misma porción de memoria compartida. *memsize* y *perm* serán ignorados.

shm_detach

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

shm_detach -- Finaliza conexión con un segmento de memoria compartida

Descripción

int **shm_detach** (int shm_identifier)

shm_detach() finaliza la conexión con la memoria compartida, especificada por el identificador *shm_identifier* creado con **shm_attach()**. Hay que recordar que la memoria compartida aún existe en el sistema Unix, y los datos aún están presentes.

shm_get_var

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

shm_get_var -- Devuelve una variable de la memoria compartida

Descripción

mixed **shm_get_var** (int id, int variable_key)

shm_get_var() devuelve la variable con la llave *variable_key* dada. La variable, queda presente en la memoria compartida.

shm_put_var

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

shm_put_var -- Inserta o actualiza una variable en la memoria compartida

Descripción

bool **shm_put_var** (int identificador_shm, int clave_variable, mixed variable)

shm_put_var() inserta o actualiza la *variable* con la *clave_variable* entregada. Todos los tipos de variables son soportados.

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Se emitirán advertencias (del nivel *E_WARNING*) si *identificador_shm* no es un índice de memoria compartida SysV, o si no existe suficiente memoria compartida disponible para completar su solicitud.

shm_remove_var

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

shm_remove_var -- Elimina una variable de la memoria compartida

Descripción

int **shm_remove_var** (int id, int variable_key)

Elimina la varible que se corresponde con la llave *variable_key* dada, liberando la memoria que ocupaba aquella.

shm_remove

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

shm_remove -- Elimina memoria compartida del sistma Unix

Descripción

int **shm_remove** (int shm_identifier)

Elimina la memoria compartida de un sistema Unix, Todos los datos serán destruídos.

CX. SESAM database functions

SESAM/SQL-Server is a mainframe database system, developed by Fujitsu Siemens Computers, Germany. It runs on high-end mainframe servers using the operating system BS2000/OSD.

In numerous productive BS2000 installations, SESAM/SQL-Server has proven ...

- the ease of use of Java-, Web- and client/server connectivity,
- the capability to work with an availability of more than 99.99%,
- the ability to manage tens and even hundreds of thousands of users.

Now there is a PHP3 SESAM interface available which allows database operations via PHP-scripts.

> **Configuration notes:** There is no standalone support for the PHP SESAM interface, it works only as an integrated Apache module. In the Apache PHP module, this SESAM interface is configured using Apache directives.
>
> **Tabla 1. SESAM Configuration directives**

Directive	Meaning
php3_sesam_oml	Name of BS2000 PLAM library containing the loadable SESAM driver modules. Required for using SESAM functions. Example: `php3_sesam_oml $.SYSLNK.SESAM-SQL.030`

php3_sesam_configfile	Name of SESAM application configuration file. Required for using SESAM functions.
	
	Example:
	`php3_sesam_configfile $SESAM.SESAM.CONF.AW`
	It will usually contain a configuration like (see SESAM reference manual):
	`CNF=B`
	`NAM=K`
	`NOTYPE`

php3_sesam_messagecatalog	Name of SESAM message catalog file. In most cases, this directive is not neccessary. Only if the SESAM message file is not installed in the system's BS2000 message file table, it can be set with this directive. Example: `php3_sesam_messagecatalog $.SYSMES.SESAM-SQL.030`

In addition to the configuration of the PHP/SESAM interface, you have to configure the SESAM-Database server itself on your mainframe as usual. That means:

- starting the SESAM database handler (DBH), and
- connecting the databases with the SESAM database handler

To get a connection between a PHP script and the database handler, the *CNF* and *NAM* parameters of the selected SESAM configuration file must match the id of the started database handler.

In case of distributed databases you have to start a SESAM/SQL-DCN agent with the distribution table including the host and database names.

The communication between PHP (running in the POSIX subsystem) and the database handler (running outside the POSIX subsystem) is realized by a special driver module called SQLSCI and SESAM connection modules using common memory. Because of the common memory access, and because PHP is a static part of the web server, database accesses are very fast, as they do not require remote accesses via ODBC, JDBC or UTM.

Only a small stub loader (SESMOD) is linked with PHP, and the SESAM connection modules are pulled in from SESAM's OML PLAM library. In the configuration, you must tell PHP the name of this PLAM library, and the file link to use for the SESAM configuration file (As of SESAM V3.0, SQLSCI is available in the SESAM Tool Library, which is part of the standard distribution).

Because the SQL command quoting for single quotes uses duplicated single quotes (as

opposed to a single quote preceded by a backslash, used in some other databases), it is advisable to set the PHP configuration directives *php3_magic_quotes_gpc* and *php3_magic_quotes_sybase* to *On* for all PHP scripts using the SESAM interface.

Runtime considerations: Because of limitations of the BS2000 process model, the driver can be loaded only after the Apache server has forked off its server child processes. This will slightly slow down the initial SESAM request of each child, but subsequent accesses will respond at full speed.

When explicitly defining a Message Catalog for SESAM, that catalog will be loaded each time the driver is loaded (i.e., at the initial SESAM request). The BS2000 operating system prints a message after successful load of the message catalog, which will be sent to Apache's error_log file. BS2000 currently does not allow suppression of this message, it will slowly fill up the log.

Make sure that the SESAM OML PLAM library and SESAM configuration file are readable by the user id running the web server. Otherwise, the server will be unable to load the driver, and will not allow to call any SESAM functions. Also, access to the database must be granted to the user id under which the Apache server is running. Otherwise, connections to the SESAM database handler will fail.

Cursor Types: The result cursors which are allocated for SQL "select type" queries can be either "sequential" or "scrollable". Because of the larger memory overhead needed by "scrollable" cursors, the default is "sequential".

When using "scrollable" cursors, the cursor can be freely positioned on the result set. For each "scrollable" query, there are global default values for the scrolling type (initialized to: *SESAM_SEEK_NEXT*) and the scrolling offset which can either be set once by **sesam_seek_row()** or each time when fetching a row using **sesam_fetch_row()**. When fetching a row using a "scrollable" cursor, the following post-processing is done for the global default values for the scrolling type and scrolling offset:

Tabla 2. Scrolled Cursor Post-Processing

Scroll Type	Action
SESAM_SEEK_NEXT	none
SESAM_SEEK_PRIOR	none
SESAM_SEEK_FIRST	set scroll type to *SESAM_SEEK_NEXT*
SESAM_SEEK_LAST	set scroll type to *SESAM_SEEK_PRIOR*
SESAM_SEEK_ABSOLUTE	Auto-Increment internal offset value
SESAM_SEEK_RELATIVE	none. (maintain global default *offset* value, which allows for, e.g., fetching each 10th row backwards)

Porting note: Because in the PHP world it is natural to start indexes at zero (rather than 1), some adaptions have been made to the SESAM interface: whenever an indexed array is starting with index 1 in the native SESAM interface, the PHP interface uses index 0 as a starting point. E.g., when retrieving columns with **sesam_fetch_row()**, the first column has the index 0, and the subsequent columns have indexes up to (but not including) the column count ($array["count"]). When porting SESAM applications from

other high level languages to PHP, be aware of this changed interface. Where appropriate, the description of the respective php sesam functions include a note that the index is zero based.

Security concerns: When allowing access to the SESAM databases, the web server user should only have as little privileges as possible. For most databases, only read access privilege should be granted. Depending on your usage scenario, add more access rights as you see fit. Never allow full control to any database for any user from the 'net! Restrict access to php scripts which must administer the database by using password control and/or SSL security.

Migration from other SQL databases: No two SQL dialects are ever 100% compatible. When porting SQL applications from other database interfaces to SESAM, some adaption may be required. The following typical differences should be noted:

- Vendor specific data types

  Some vendor specific data types may have to be replaced by standard SQL data types (e.g., *TEXT* could be replaced by *VARCHAR(max. size)*).

- Keywords as SQL identifiers

  In SESAM (as in standard SQL), such identifiers must be enclosed in double quotes (or renamed).

- Display length in data types

  SESAM data types have a precision, not a display length. Instead of *int(4)* (intended use: integers up to '9999'), SESAM requires simply *int* for an implied size of 31 bits. Also, the only datetime data types available in SESAM are: *DATE*, *TIME(3)* and *TIMESTAMP(3)*.

- SQL types with vendor-specific *unsigned*, *zerofill*, or *auto_increment* attributes

  Unsigned and *zerofill* are not supported. *Auto_increment* is automatic (use *"INSERT ... VALUES(*, ...)"* instead of *"... VALUES(0, ...)"* to take advantage of SESAM-implied auto-increment.

- **int ... DEFAULT '0000'**

  Numeric variables must not be initialized with string constants. Use **DEFAULT 0** instead. To initialize variables of the datetime SQL data types, the initialization string must be prefixed with the respective type keyword, as in: *CREATE TABLE exmpl (xtime timestamp(3) DEFAULT TIMESTAMP '1970-01-01 00:00:00.000' NOT* ***NULL****);*

- **$count = xxxx_num_rows();**

  Some databases promise to guess/estimate the number of the rows in a query result, even though the returned value is grossly incorrect. SESAM does not know the number of rows in a query result before actually fetching them. If you REALLY need the count, try **SELECT COUNT(...) WHERE ...**, it will tell you the number of hits. A second query will (hopefully) return the results.

- **DROP TABLE thename;**

  In SESAM, in the **DROP TABLE** command, the table name must be either followed by the keyword *RESTRICT* or *CASCADE*. When specifying *RESTRICT*, an error is returned if there are dependent objects (e.g., VIEWs), while with *CASCADE*, dependent objects will be deleted along with the specified table.

Notes on the use of various SQL types: SESAM does not currently support the BLOB type. A future version of SESAM will have support for BLOB.

At the PHP interface, the following type conversions are automatically applied when retrieving SQL fields:

Tabla 3. SQL to PHP Type Conversions

SQL Type	PHP Type
SMALLINT, INTEGER	"integer"
NUMERIC, DECIMAL, FLOAT, REAL, DOUBLE	"double"
DATE, TIME, TIMESTAMP	"string"
VARCHAR, CHARACTER	"string"

When retrieving a complete row, the result is returned as an array. Empty fields are not filled in, so you will have to check for the existence of the individual fields yourself (use **isset()** or **empty()** to test for empty fields). That allows more user control over the appearance of empty fields (than in the case of an empty string as the representation of an empty field).

Support of SESAM's "multiple fields" feature: The special "multiple fields" feature of SESAM allows a column to consist of an array of fields. Such a "multiple field" column can be created like this:

Ejemplo 1. Creating a "multiple field" column

```
CREATE TABLE multi_field_test (
    pkey CHAR(20) PRIMARY KEY,
    multi(3) CHAR(12)
)
```

and can be filled in using:

Ejemplo 2. Filling a "multiple field" column

```
INSERT INTO multi_field_test (pkey, multi(2..3) )
    VALUES ('Second', <'first_val', 'second_val'>)
```

Note that (like in this case) leading empty sub-fields are ignored, and the filled-in values are collapsed, so that in the above example the result will appear as multi(1..2) instead of multi(2..3).

When retrieving a result row, "multiple columns" are accessed like "inlined" additional columns. In the example above, "pkey" will have the index 0, and the three "multi(1..3)" columns will be accessible as indices 1 through 3.

For specific SESAM details, please refer to the SESAM/SQL-Server documentation (english) or the SESAM/SQL-Server documentation (german), both available online, or use the respective manuals.

Tabla de contenidos

sesam_affected_rows

(PHP 3 CVS only)

sesam_affected_rows -- Get number of rows affected by an immediate query

Description

int **sesam_affected_rows** (string result_id)

result_id is a valid result id returned by **sesam_query()**.

Returns the number of rows affected by a query associated with *result_id*.

The **sesam_affected_rows()** function can only return useful values when used in combination with "immediate" SQL statements (updating operations like *INSERT*, *UPDATE* and *DELETE*) because SESAM does not deliver any "affected rows" information for "select type" queries. The number returned is the number of affected rows.

See also: **sesam_query()** and **sesam_execimm()**

```
$result = sesam_execimm ("DELETE FROM PHONE WHERE LASTNAME = '".strtoupper ($name)."'");
if (!$result) {
    ... error ...
}
print sesam_affected_rows ($result).
    " entries with last name ".$name." deleted.\n"
```

sesam_commit

(PHP 3 CVS only)

sesam_commit -- Commit pending updates to the SESAM database

Description

bool **sesam_commit** (void)

Returns: **TRUE** on success, **FALSE** on errors

sesam_commit() commits any pending updates to the database.

Note that there is no "auto-commit" feature as in other databases, as it could lead to accidental data loss. Uncommitted data at the end of the current script (or when calling **sesam_disconnect()**) will be discarded by an implied **sesam_rollback()** call.

See also: **sesam_rollback()**.

Ejemplo 1. Committing an update to the SESAM database

```
<?php
if (sesam_connect ("mycatalog", "myschema", "otto")) {
    if (!sesam_execimm ("INSERT INTO mytable VALUES (*, 'Small Test', <0, 8, 15>)"))
        die("insert failed");
    if (!sesam_commit())
        die("commit failed");
}
?>
```

sesam_connect

(PHP 3 CVS only)

sesam_connect -- Open SESAM database connection

Description

bool **sesam_connect** (string catalog, string schema, string user)

Returns **TRUE** if a connection to the SESAM database was made, or **FALSE** on error.

sesam_connect() establishes a connection to an SESAM database handler task. The connection is always "persistant" in the sense that only the very first invocation will actually load the driver from the configured SESAM OML PLAM library. Subsequent calls will reuse the driver and will immediately use the given catalog, schema, and user.

When creating a database, the *"catalog"* name is specified in the SESAM configuration directive // **ADD-SQL-DATABASE-CATALOG-LIST ENTRY-1 = *CATALOG(CATALOG-NAME = catalogname,...)**

The *"schema"* references the desired database schema (see SESAM handbook).

The *"user"* argument references one of the users which are allowed to access this *"catalog"* / *"schema"* combination. Note that *"user"* is completely independent from both the system's user id's and from HTTP user/password protection. It appears in the SESAM configuration only.

See also **sesam_disconnect()**.

Ejemplo 1. Connect to a SESAM database

```
<?php
if (!sesam_connect ("mycatalog", "myschema", "otto")
    die("Unable to connect to SESAM";
?>
```

sesam_diagnostic

(PHP 3 CVS only)

sesam_diagnostic -- Return status information for last SESAM call

Description

array **sesam_diagnostic** (void)

Returns an associative array of status and return codes for the last SQL query/statement/command. Elements of the array are:

Tabla 1. Status information returned by sesam_diagnostic()

Element	Contents
$array ["sqlstate"]	5 digit SQL return code (see the SESAM manual for the description of the possible values of SQLSTATE)
$array ["rowcount"]	number of affected rows in last update/insert/delete (set after "immediate" statements only)
$array["errmsg"]	"human readable" error message string (set after errors only)
$array["errcol"]	error column number of previous error (0-based; or -1 if undefined. Set after errors only)
$array["errlin"]	error line number of previous error (0-based; or -1 if undefined. Set after errors only)

In the following example, a syntax error (E SEW42AE ILLEGAL CHARACTER) is displayed by including the offending SQL statement and pointing to the error location:

Ejemplo 1. Displaying SESAM error messages with error position

```
<?php
// Function which prints a formatted error message,
// displaying a pointer to the syntax error in the
// SQL statement
function PrintReturncode ($exec_str) {
    $err = Sesam_Diagnostic();
    $colspan=4; // 4 cols for: sqlstate, errlin, errcol, rowcount
    if ($err["errlin"] == -1)
        --$colspan;
    if ($err["errcol"] == -1)
        --$colspan;
    if ($err["rowcount"] == 0)
        --$colspan;
    echo "<TABLE BORDER>\n";
    echo "<TR><TH COLSPAN=".$colspan."><FONT COLOR=red>ERROR:</FONT> ".
                htmlspecialchars($err["errmsg"])."</TH></TR>\n";
    if ($err["errcol"] >= 0) {
        echo "<TR><TD COLSPAN=".$colspan."><PRE>\n";
        $errstmt = $exec_str."\n";
        for ($lin=0; $errstmt != ""; ++$lin) {
            if ($lin != $err["errlin"]) { // $lin is less or greater than errlin
                if (!($i = strchr ($errstmt, "\n")))
                    $i = "";
                $line = substr ($errstmt, 0, strlen($errstmt)-strlen($i)+1);
                $errstmt = substr($i, 1);
                if ($line != "\n")
                    print htmlspecialchars ($line);
            } else {
                if (! ($i = strchr ($errstmt, "\n")))
                    $i = "";
                $line = substr ($errstmt, 0, strlen ($errstmt)-strlen($i)+1);
                $errstmt = substr($i, 1);
                for ($col=0; $col < $err["errcol"]; ++$col)
                    echo (substr($line, $col, 1) == "\t") ? "\t" : ".";
                echo "<FONT COLOR=RED><BLINK>\\</BLINK></FONT>\n";
                print "<FONT COLOR=\"#880000\">".htmlspecialchars($line)."</FONT>";
                for ($col=0; $col < $err["errcol"]; ++$col)
                    echo (substr ($line, $col, 1) == "\t") ? "\t" : ".";
                echo "<FONT COLOR=RED><BLINK>/</BLINK></FONT>\n";
            }
        }
        echo "</PRE></TD></TR>\n";
    }
    echo "<TR>\n";
    echo " <TD>sqlstate=" . $err["sqlstate"] . "</TD>\n";
    if ($err["errlin"] != -1)
        echo " <TD>errlin=" . $err["errlin"] . "</TD>\n";
    if ($err["errcol"] != -1)
        echo " <TD>errcol=" . $err["errcol"] . "</TD>\n";
    if ($err["rowcount"] != 0)
         echo " <TD>rowcount=" . $err["rowcount"] . "</TD>\n";
    echo "</TR>\n";
    echo "</TABLE>\n";
}

if (!sesam_connect ("mycatalog", "phoneno", "otto"))
  die ("cannot connect");

$stmt = "SELECT * FROM phone\n".
        " WHERE@ LASTNAME='KRAEMER'\n".
        " ORDER BY FIRSTNAME";
if (!($result = sesam_query ($stmt)))
    PrintReturncode ($stmt);
?>
```

See also: **sesam_errormsg()** for simple access to the error string only

sesam_disconnect

(PHP 3 CVS only)

sesam_disconnect -- Detach from SESAM connection

Description

bool **sesam_disconnect** (void)

Returns: always **TRUE**.

sesam_disconnect() closes the logical link to a SESAM database (without actually disconnecting and unloading the driver).

Note that this isn't usually necessary, as the open connection is automatically closed at the end of the script's execution. Uncommitted data will be discarded, because an implicit **sesam_rollback()** is executed.

sesam_disconnect() will not close the persistent link, it will only invalidate the currently defined *"catalog"*, *"schema"* and *"user"* triple, so that any sesam function called after **sesam_disconnect()** will fail.

See also: **sesam_connect()**.

Ejemplo 1. Closing a SESAM connection

```
if (sesam_connect ("mycatalog", "myschema", "otto")) {
    ... some queries and stuff ...
    sesam_disconnect();
}
```

sesam_errormsg

(PHP 3 CVS only)

sesam_errormsg -- Returns error message of last SESAM call

Description

string **sesam_errormsg** (void)

Returns the SESAM error message associated with the most recent SESAM error.

```
if (!sesam_execimm ($stmt))
  printf ("%s<br>\n", sesam_errormsg());
```

See also: **sesam_diagnostic()** for the full set of SESAM SQL status information

sesam_execimm

(PHP 3 CVS only)

sesam_execimm -- Execute an "immediate" SQL-statement

Description

string **sesam_execimm** (string query)

Returns: A SESAM "result identifier" on success, or **FALSE** on error.

sesam_execimm() executes an "immediate" statement (i.e., a statement like UPDATE, INSERT or DELETE which returns no result, and has no INPUT or OUTPUT variables). "select type" queries can not be used with **sesam_execimm()**. Sets the *affected_rows* value for retrieval by the **sesam_affected_rows()** function.

Note that **sesam_query()** can handle both "immediate" and "select-type" queries. Use **sesam_execimm()** only if you know beforehand what type of statement will be executed. An attempt to use SELECT type queries with **sesam_execimm()** will return *$err["sqlstate"] == "42SBW"*.

The returned "result identifier" can not be used for retrieving anything but the **sesam_affected_rows()**; it is only returned for symmetry with the **sesam_query()** function.

```
$stmt = "INSERT INTO mytable VALUES ('one', 'two')";
$result = sesam_execimm ($stmt);
$err = sesam_diagnostic();
print ("sqlstate = ".$err["sqlstate"]."\n".
       "Affected rows = ".$err["rowcount"]." == ".
       sesam_affected_rows($result)."\n");
```

See also: **sesam_query()** and **sesam_affected_rows()**.

sesam_fetch_array

(PHP 3 CVS only)

sesam_fetch_array -- Fetch one row as an associative array

Description

array **sesam_fetch_array** (string result_id [, int whence [, int offset]])

Returns an array that corresponds to the fetched row, or **FALSE** if there are no more rows.

sesam_fetch_array() is an alternative version of **sesam_fetch_row()**. Instead of storing the data in the numeric indices of the result array, it stores the data in associative indices, using the field names as keys.

result_id is a valid result id returned by **sesam_query()** (select type queries only!).

For the valid values of the optional *whence*and *offset* parameters, see the **sesam_fetch_row()** function for details.

sesam_fetch_array() fetches one row of data from the result associated with the specified result identifier. The row is returned as an associative array. Each result column is stored with an associative index equal to its column (aka. field) name. The column names are converted to lower

case.

Columns without a field name (e.g., results of arithmetic operations) and empty fields are not stored in the array. Also, if two or more columns of the result have the same column names, the later column will take precedence. In this situation, either call **sesam_fetch_row()** or make an alias for the column.

```
SELECT TBL1.COL AS FOO, TBL2.COL AS BAR FROM TBL1, TBL2
```

A special handling allows fetching "multiple field" columns (which would otherwise all have the same column names). For each column of a "multiple field", the index name is constructed by appending the string "(n)" where n is the sub-index of the multiple field column, ranging from 1 to its declared repetition factor. The indices are NOT zero based, in order to match the nomenclature used in the respective query syntax. For a column declared as:

```
CREATE TABLE ... ( ... MULTI(3) INT )
```

the associative indices used for the individual "multiple field" columns would be *"multi(1)"*, *"multi (2)"*, and *"multi(3)"* respectively.

Subsequent calls to **sesam_fetch_array()** would return the next (or prior, or n'th next/prior, depending on the scroll attributes) row in the result set, or **FALSE** if there are no more rows.

Ejemplo 1. SESAM fetch array

```
<?php
$result = sesam_query ("SELECT * FROM phone\n".
                       "  WHERE LASTNAME='".strtoupper($name)."'\n".
                       "  ORDER BY FIRSTNAME", 1);
if (!$result) {
    ... error ...
}
// print the table:
print "<TABLE BORDER>\n";
while (($row = sesam_fetch_array ($result)) && count ($row) > 0) {
    print " <TR>\n";
    print " <TD>".htmlspecialchars ($row["firstname"])."</TD>\n";
    print " <TD>".htmlspecialchars ($row["lastname"])."</TD>\n";
    print " <TD>".htmlspecialchars ($row["phoneno"])."</TD>\n";
    print " </TR>\n";
}
print "</TABLE>\n";
sesam_free_result ($result);
?>
```

See also: **sesam_fetch_row()** which returns an indexed array.

sesam_fetch_result

(PHP 3 CVS only)

sesam_fetch_result -- Return all or part of a query result

Description

mixed **sesam_fetch_result** (string result_id [, int max_rows])

Returns a mixed array with the query result entries, optionally limited to a maximum of *max_rows* rows. Note that both row and column indexes are zero-based.

Tabla 1. Mixed result set returned by sesam_fetch_result()

Array Element	Contents
int $arr ["count"]	number of columns in result set (or zero if this was an "immediate" query)
int $arr ["rows"]	number of rows in result set (between zero and *max_rows*)
bool $arr ["truncated"]	**TRUE** if the number of rows was at least *max_rows*, **FALSE** otherwise. Note that even when this is **TRUE**, the next **sesam_fetch_result()** call may return zero rows because there are no more result entries.
mixed $arr [col][row]	result data for all the fields at row(*row*) and column(*col*), (where the integer index *row* is between 0 and *$arr["rows"]-1*, and *col* is between 0 and *$arr["count"]-1*). Fields may be empty, so you must check for the existence of a field by using the php **isset()** function. The type of the returned fields depend on the respective SQL type declared for its column (see SESAM overview for the conversions applied). SESAM "multiple fields" are "inlined" and treated like a sequence of columns.

Note that the amount of memory used up by a large query may be gigantic. Use the *max_rows* parameter to limit the maximum number of rows returned, unless you are absolutely sure that your result will not use up all available memory.

See also: **sesam_fetch_row()**, and **sesam_field_array()** to check for "multiple fields". See the description of the **sesam_query()** function for a complete example using **sesam_fetch_result()**.

sesam_fetch_row

(PHP 3 CVS only)

sesam_fetch_row -- Fetch one row as an array

Description

array **sesam_fetch_row** (string result_id [, int whence [, int offset]])

Returns an array that corresponds to the fetched row, or **FALSE** if there are no more rows.

The number of columns in the result set is returned in an associative array element $array["count"]. Because some of the result columns may be empty, the **count()** function can not be used on the result row returned by **sesam_fetch_row()**.

result_id is a valid result id returned by **sesam_query()** (select type queries only!).

whence is an optional parameter for a fetch operation on "scrollable" cursors, which can be set to the following predefined constants:

Tabla 1. Valid values for *"whence"* parameter

Value	Constant	Meaning
0	*SESAM_SEEK_NEXT*	read sequentially (after fetch, the internal default is set to *SESAM_SEEK_NEXT*)
1	*SESAM_SEEK_PRIOR*	read sequentially backwards (after fetch, the internal default is set to *SESAM_SEEK_PRIOR*)
2	*SESAM_SEEK_FIRST*	rewind to first row (after fetch, the default is set to *SESAM_SEEK_NEXT*)
3	*SESAM_SEEK_LAST*	seek to last row (after fetch, the default is set to *SESAM_SEEK_PRIOR*)
4	*SESAM_SEEK_ABSOLUTE*	seek to absolute row number given as *offset* (Zero-based. After fetch, the internal default is set to *SESAM_SEEK_ABSOLUTE*, and the internal offset value is auto-incremented)
5	*SESAM_SEEK_RELATIVE*	seek relative to current scroll position, where *offset* can be a positive or negative offset value.

This parameter is only valid for "scrollable" cursors.

When using "scrollable" cursors, the cursor can be freely positioned on the result set. If the *whence* parameter is omitted, the global default values for the scrolling type (initialized to: *SESAM_SEEK_NEXT*, and settable by **sesam_seek_row()**) are used. If *whence* is supplied, its value replaces the global default.

offset is an optional parameter which is only evaluated (and required) if *whence* is either *SESAM_SEEK_RELATIVE* or *SESAM_SEEK_ABSOLUTE*. This parameter is only valid for "scrollable" cursors.

sesam_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array (indexed by values between *0* and *$array["count"]-1*). Fields may be empty, so you must check for the existence of a field by using the php **isset()** function. The type of the returned fields depend on the respective SQL type declared for its column (see SESAM overview for the conversions applied). SESAM "multiple fields" are "inlined" and treated like a sequence of columns.

Subsequent calls to **sesam_fetch_row()** would return the next (or prior, or n'th next/prior, depending on the scroll attributes) row in the result set, or **`FALSE`** if there are no more rows.

Ejemplo 1. SESAM fetch rows

```
<?php
$result = sesam_query ("SELECT * FROM phone\n".
                       "  WHERE LASTNAME='".strtoupper($name)."'\n".
                       "  ORDER BY FIRSTNAME", 1);
if (!$result) {
    ... error ...
}
// print the table in backward order
print "<TABLE BORDER>\n";
$row = sesam_fetch_row ($result, SESAM_SEEK_LAST);
while (is_array ($row)) {
    print " <TR>\n";
    for ($col = 0; $col < $row["count"]; ++$col) {
        print "  <TD>".htmlspecialchars ($row[$col])."</TD>\n";
    }
    print " </TR>\n";
    // use implied SESAM_SEEK_PRIOR
    $row = sesam_fetch_row ($result);
}
print "</TABLE>\n";
sesam_free_result ($result);
?>
```

See also: **sesam_fetch_array()** which returns an associative array, and **sesam_fetch_result()** which returns many rows per invocation.

sesam_field_array

(PHP 3 CVS only)

sesam_field_array -- Return meta information about individual columns in a result

Description

array **sesam_field_array** (string result_id)

result_id is a valid result id returned by **sesam_query()**.

Returns a mixed associative/indexed array with meta information (column name, type, precision, ...) about individual columns of the result after the query associated with *result_id*.

Tabla 1. Mixed result set returned by sesam_field_array()

Array Element	Contents
int $arr ["count"]	Total number of columns in result set (or zero if this was an "immediate" query). SESAM "multiple fields" are "inlined" and treated like the respective number of columns.
string $arr [col] ["name"]	column name for column(*col*), where *col* is between 0 and *$arr["count"]-1*. The returned value can be the empty string (for dynamically computed columns). SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same column name.

Array Element	Contents
string $arr [col] ["count"]	The "count" attribute describes the repetition factor when the column has been declared as a "multiple field". Usually, the "count" attribute is 1. The first column of a "multiple field" column however contains the number of repetitions (the second and following column of the "multiple field" contain a "count" attribute of 1). This can be used to detect "multiple fields" in the result set. See the example shown in the **sesam_query()** description for a sample use of the "count" attribute.
string $arr [col]["type"]	php variable type of the data for column(*col*), where *col* is between 0 and *$arr ["count"]-1*. The returned value can be one of • "integer" • "double" • "string" depending on the SQL type of the result. SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same php type.
string $arr [col] ["sqltype"]	SQL variable type of the column data for column(*col*), where *col* is between 0 and *$arr["count"]-1*. The returned value can be one of • "CHARACTER" • "VARCHAR" • "NUMERIC" • "DECIMAL" • "INTEGER" • "SMALLINT" • "FLOAT" • "REAL" • "DOUBLE" • "DATE" • "TIME" • "TIMESTAMP" describing the SQL type of the result. SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same SQL type.

Array Element	Contents
string $arr [col] ["length"]	The SQL "length" attribute of the SQL variable in column(*col*), where *col* is between 0 and *$arr["count"]-1*. The "length" attribute is used with "CHARACTER" and "VARCHAR" SQL types to specify the (maximum) length of the string variable. SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same length attribute.
string $arr [col] ["precision"]	The "precision" attribute of the SQL variable in column(*col*), where *col* is between 0 and *$arr["count"]-1*. The "precision" attribute is used with numeric and time data types. SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same precision attribute.
string $arr [col]["scale"]	The "scale" attribute of the SQL variable in column(*col*), where *col* is between 0 and *$arr["count"]-1*. The "scale" attribute is used with numeric data types. SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same scale attribute.

See the **sesam_query()** function for an example of the **sesam_field_array()** use.

sesam_field_name

(PHP 3 CVS only)

sesam_field_name -- Return one column name of the result set

Description

int **sesam_field_name** (string result_id, int index)

Returns the name of a field (i.e., the column name) in the result set, or **`FALSE`** on error.

For "immediate" queries, or for dynamic columns, an empty string is returned.

> **Nota:** The column index is zero-based, not one-based as in SESAM.

See also: **sesam_field_array()**. It provides an easier interface to access the column names and types, and allows for detection of "multiple fields".

sesam_free_result

(PHP 3 CVS only)

sesam_free_result -- Releases resources for the query

Description

int **sesam_free_result** (string result_id)

Releases resources for the query associated with *result_id*. Returns **`FALSE`** on error.

sesam_num_fields

(PHP 3 CVS only)

sesam_num_fields -- Return the number of fields/columns in a result set

Description

int **sesam_num_fields** (string result_id)

After calling **sesam_query()** with a "select type" query, this function gives you the number of columns in the result. Returns an integer describing the total number of columns (aka. fields) in the current *result_id* result set or **FALSE** on error.

For "immediate" statements, the value zero is returned. The SESAM "multiple field" columns count as their respective dimension, i.e., a three-column "multiple field" counts as three columns.

See also: **sesam_query()** and **sesam_field_array()** for a way to distinguish between "multiple field" columns and regular columns.

sesam_query

(PHP 3 CVS only)

sesam_query -- Perform a SESAM SQL query and prepare the result

Description

string **sesam_query** (string query [, bool scrollable])

Returns: A SESAM "result identifier" on success, or **FALSE** on error.

A "result_id" resource is used by other functions to retrieve the query results.

sesam_query() sends a query to the currently active database on the server. It can execute both "immediate" SQL statements and "select type" queries. If an "immediate" statement is executed, then no cursor is allocated, and any subsequent **sesam_fetch_row()** or **sesam_fetch_result()** call will return an empty result (zero columns, indicating end-of-result). For "select type" statements, a result descriptor and a (scrollable or sequential, depending on the optional boolean *scrollable* parameter) cursor will be allocated. If *scrollable* is omitted, the cursor will be sequential.

When using "scrollable" cursors, the cursor can be freely positioned on the result set. For each "scrollable" query, there are global default values for the scrolling type (initialized to: *SESAM_SEEK_NEXT*) and the scrolling offset which can either be set once by **sesam_seek_row()** or each time when fetching a row using **sesam_fetch_row()**.

For "immediate" statements, the number of affected rows is saved for retrieval by the **sesam_affected_rows()** function.

See also: **sesam_fetch_row()** and **sesam_fetch_result()**.
Ejemplo 1. Show all rows of the "phone" table as a html table

```
<?php
if (!sesam_connect ("phonedb", "demo", "otto"))
    die ("cannot connect");
$result = sesam_query ("select * from phone");
if (!$result) {
    $err = sesam_diagnostic();
    die ($err["errmsg"]);
}
echo "<TABLE BORDER>\n";
// Add title header with column names above the result:
if ($cols = sesam_field_array ($result)) {
    echo " <TR><TH COLSPAN=".$cols["count"].">Result:</TH></TR>\n";
    echo " <TR>\n";
    for ($col = 0; $col < $cols["count"]; ++$col) {
        $colattr = $cols[$col];
        /* Span the table head over SESAM's "Multiple Fields": */
        if ($colattr["count"] > 1) {
            echo "  <TH COLSPAN=".$colattr["count"].">".$colattr["name"].
                "(1..".$colattr["count"].")</TH>\n";
            $col += $colattr["count"] - 1;
        } else
            echo "  <TH>" . $colattr["name"] . "</TH>\n";
    }
    echo " </TR>\n";
}

do {
    // Fetch the result in chunks of 100 rows max.
    $ok = sesam_fetch_result ($result, 100);
    for ($row=0; $row < $ok["rows"]; ++$row) {
        echo " <TR>\n";
        for ($col = 0; $col < $ok["cols"]; ++$col) {
            if (isset($ok[$col][$row]))
                echo "  <TD>" . $ok[$col][$row] . "</TD>\n";
            } else {
                echo "  <TD>-empty-</TD>\n";
            }
        }
        echo " </TR>\n";
    }
}
while ($ok["truncated"]) { // while there may be more data
    echo "</TABLE>\n";
}
// free result id
sesam_free_result($result);
?>
```

sesam_rollback

(PHP 3 CVS only)

sesam_rollback -- Discard any pending updates to the SESAM database

Description

bool **sesam_rollback** (void)

Returns: **TRUE** on success, **FALSE** on errors

sesam_rollback() discards any pending updates to the database. Also affected are result cursors and result descriptors.

At the end of each script, and as part of the **sesam_disconnect()** function, an implied **sesam_rollback()** is executed, discarding any pending changes to the database.

See also: **sesam_commit()**.

Ejemplo 1. Discarding an update to the SESAM database

```
<?php
if (sesam_connect ("mycatalog", "myschema", "otto")) {
    if (sesam_execimm ("INSERT INTO mytable VALUES (*, 'Small Test', <0, 8, 15>)")
        && sesam_execimm ("INSERT INTO othertable VALUES (*, 'Another Test', 1)"))
        sesam_commit();
    else
        sesam_rollback();
}
?>
```

sesam_seek_row

(PHP 3 CVS only)

sesam_seek_row -- Set scrollable cursor mode for subsequent fetches

Description

bool **sesam_seek_row** (string result_id, int whence [, int offset])

result_id is a valid result id (select type queries only, and only if a "scrollable" cursor was requested when calling **sesam_query()**).

whence sets the global default value for the scrolling type, it specifies the scroll type to use in subsequent fetch operations on "scrollable" cursors, which can be set to the following predefined constants:

Tabla 1. Valid values for *"whence"* parameter

Value	Constant	Meaning
0	*SESAM_SEEK_NEXT*	read sequentially
1	*SESAM_SEEK_PRIOR*	read sequentially backwards
2	*SESAM_SEEK_FIRST*	fetch first row (after fetch, the default is set to *SESAM_SEEK_NEXT*)
3	*SESAM_SEEK_LAST*	fetch last row (after fetch, the default is set to *SESAM_SEEK_PRIOR*)
4	*SESAM_SEEK_ABSOLUTE*	fetch absolute row number given as *offset* (Zero-based. After fetch, the default is set to *SESAM_SEEK_ABSOLUTE*, and the offset value is auto-incremented)
5	*SESAM_SEEK_RELATIVE*	fetch relative to current scroll position, where *offset* can be a positive or negative offset value (this also sets the default "offset" value for subsequent fetches).

offset is an optional parameter which is only evaluated (and required) if *whence* is either *SESAM_SEEK_RELATIVE* or *SESAM_SEEK_ABSOLUTE*.

sesam_settransaction

(PHP 3 CVS only)

sesam_settransaction -- Set SESAM transaction parameters

Description

bool **sesam_settransaction** (int isolation_level, int read_only)

Returns: **TRUE** if the values are valid, and the **settransaction()** operation was successful, **FALSE** otherwise.

sesam_settransaction() overrides the default values for the "isolation level" and "read-only" transaction parameters (which are set in the SESAM configuration file), in order to optimize subsequent queries and guarantee database consistency. The overridden values are used for the next transaction only.

sesam_settransaction() can only be called before starting a transaction, not after the transaction has been started already.

To simplify the use in php scripts, the following constants have been predefined in php (see SESAM handbook for detailed explanation of the semantics):

Tabla 1. Valid values for *"Isolation_Level"* parameter

Value	Constant	Meaning
1	*SESAM_TXISOL_READ_UNCOMMITTED*	Read Uncommitted
2	*SESAM_TXISOL_READ_COMMITTED*	Read Committed
3	*SESAM_TXISOL_REPEATABLE_READ*	Repeatable Read
4	*SESAM_TXISOL_SERIALIZABLE*	Serializable

Tabla 2. Valid values for *"Read_Only"* parameter

Value	Constant	Meaning
0	*SESAM_TXREAD_READWRITE*	Read/Write
1	*SESAM_TXREAD_READONLY*	Read-Only

The values set by **sesam_settransaction()** will override the default setting specified in the SESAM configuration file.

Ejemplo 1. Setting SESAM transaction parameters

```
<?php
sesam_settransaction (SESAM_TXISOL_REPEATABLE_READ,
                      SESAM_TXREAD_READONLY);
?>
```

CXI. Funciones para el manejo de sesiones

El apoyo que PHP proporciona para las sesiones consiste en una forma de conservar ciertos datos a lo largo de los subsiguientes accesos, lo cual le permite construir aplicaciones más personalizadas e incrementar el atractivo de su sitio web.

Si ya está familiarizado con el tratamiento de sesiones de PHPLIB, notará que algunos conceptos son similares al soporte de las sesiones de PHP.

A cada visitante que accede a su web se le asigna un identificador único, llamado "session id" (identificador de sesión). Éste se almacena en una cookie por parte del usuario o se propaga en la URL.

El soporte de las sesiones le permite registrar un número arbitrario de variables que se conservarán en las siguientes peticiones. Cuando un visitante acceda a su web, PHP comprobará automáticamente (si session.auto_start está puesto a 1) o cuando usted lo especifique (de forma explícita mendiante **session_start()** o implícita a través de **session_register()**) si se le ha enviado un "session id" específico con su petición, en cuyo caso se recrean las variables que se habían guardado anteriormente.

Todas las variables registradas son almacenadas tras finalizar la petición. Las variables que están indefinidas se marcan como no definidas. En los subsiguientes accesos, no estarán definidas por el módulo de sesiones a menos que el usuario las defina más tarde.

Las opciones de configuración *track_vars* y *register_globals* influyen notablemente en la forma en que las variables de la sesión se almacenan y restauran.

> **Nota:** A partir de PHP 4.0.3, *track_vars* siempre está activado.
>
> **Nota:** A partir de PHP 4.1.0, *$_SESSION* está disponible como variable global, al igual que *$_POST*, *$_GET*, *$_REQUEST* y demás. Al contrario que *$HTTP_SESSION_VARS*, *$_SESSION* siempre es global. Por tanto, no se debe usar global para *$_SESSION*.

Si *track_vars* está activado y *register_globals* está desactivado, sólo los miembros del vector asociativo global *$HTTP_SESSION_VARS* pueden ser registrados como variables de la sesión. Las variables restauradas de la sesión sólo estarán disponibles en el vector *$HTTP_SESSION_VARS*.

Ejemplo 1. Registrar una variable con *track_vars* activado

```
<?php
session_start();
if (isset($HTTP_SESSION_VARS['count'])) {
   $HTTP_SESSION_VARS['count']++;
}
else {
   $HTTP_SESSION_VARS['count'] = 0;
}
?>
```

Se recomienda usar *$_SESSION* (o *$HTTP_SESSION_VARS* con PHP 4.0.6 o inferior) por seguridad y para hacer el código más legible. Con *$_SESSION* o *$HTTP_SESSION_VARS*, no es necesario usar las funciones session_register() / session_unregister() / session_is_registered(). Los usuarios pueden acceder a una variable de la sesión como si se tratase de una variable normal.

Ejemplo 2. Registrar una variable con $_SESSION.

```
<?php
session_start();
// Use $HTTP_SESSION_VARS con PHP 4.0.6 o inferior
if (!isset($_SESSION['count'])) {
    $_SESSION['count'] = 0;
} else {
    $_SESSION['count']++;
}
?>
```

Ejemplo 3. Borrar una variable con $_SESSION.

```
<?php
session_start();
// Use $HTTP_SESSION_VARS con PHP 4.0.6 o inferior
unset($_SESSION['count']);
?>
```

Si *register_globals* está activado, todas las variables globales pueden ser registradas como variables de la sesión, y las variables de la sesión serán restauradas a sus correspondientes variables globales. Como PHP debe saber qué variables globles están registradas como variables de la sesión, los usuarios deben registrar las variables con la función session_register(), mientras que con *$HTTP_SESSION_VARS*/*$_SESSION* no es necesario usar session_register().

Atención
Si está usando *$HTTP_SESSION_VARS*/*$_SESSION* y desactiva *register_globals*, no use **session_register()**, **session_is_registered()** ni **session_unregister()**. Si activa *register_globals*, **session_unregister()** debería ser usado a partir de que las variables de la sesión sean registradas como variables globales cuando los datos de la sesión se guardan. Se recomienda desactivar *register_globals* por motivos de seguridad y rendimiendo.

Ejemplo 4. Registrar una variable con *register_globals* activado

```
<?php
if (!session_is_registered('count')) {
    session_register("count");
    $count = 0;
}
else {
    $count++;
}
?>
```

Si *track_vars* y *register_globals* están activados, las variables globales y las entradas de *$HTTP_SESSION_VARS*/*$_SESSION* harán referencia al mismo valor para variables ya registradas.

Si el usuario utiliza session_register() para registrar una variable, el vector *$HTTP_SESSION_VARS*/*$_SESSION* no contendrá esa variable hasta que se cargue de los datos de la sesión. (p.ej. hasta la próxima petición).

Hay dos formas de propagar un "session id":

- Cookies
- Parámetro en la URL

El módulo de sesiones admite ambas formas. Las Cookies son la mejor opción, pero como no son fiables (los clientes no están obligados a aceptarlas), no podemos confiar en ellas. El segundo método incrusta el "session id" directamente en las URLs.

PHP es capaz de hacerlo de forma transparente al usuario cuando se compila con *--enable-trans-sid*. Si activa esta opción, las URIs relativas serán modificadas de forma que contengan el session id

automáticamente. Alternativamente, puede usar la constante *SID* que está definida, si el cliente no envía la cookie adecuada. El *SID* puede tener la forma de *nombre_de_sesion=session_id* o ser una cadena vacía.

El ejemplo siguiente demuestra cómo registrar una variable, y cómo colocar correctamente un enlace a otra página usando la constante SID.

Ejemplo 5. Contar el número de impresiones de un usuario

```
<?php
if (!session_is_registered('count')) {
    session_register('count');
    $count = 1;
}
else {
    $count++;
}
?>

Hola, visitante. Has visto esta p&aacute;gina <?php echo $count; ?> veces.

<?php
# el <?php echo SID?> (Se puede usar <?=SID?> si short tag est&aacute; activado)
# es necesario para conservar el session id
# en caso de que el usuario haya desactivado las cookies
?>

Para continuar, haga click <A HREF="nextpage.php?<?php echo SID?>">aqu&iacute;</A>.
```

El *<?=SID?>* no es necesario si se ha usado *--enable-trans-sid* al compilar PHP.

> **Nota:** Se asume que las URLs no relativas apuntan a sitios web externos, y por tanto no se añade el SID, ya que pasar el SID a un servidor diferente podría ocasionar un agujero de seguridad.

Para implementar el almacenamiento en bases de datos o en otro tipo de almacenamiento, necesitará usar **session_set_save_handler()()** para crear una colección de funciones de almacenamiento a nivel de usuario.

El sistema de control de sesiones soporta varias opciones de configuración que puede colocar en su archivo `php.ini`. Les daremos un pequeño repaso.

- *session.save_handler* define el nombre del controlador que se usa para almacenar y recuperar los datos asociados a la sesión. Su valor por defecto es *files*.
- *session.save_path* define el argumento que se pasa al controlador de almacenamiento. Si elige el controlador de archivos por defecto, esta es la ruta donde los archivos se crean. Por defecto es */tmp*. Si la profundidad de la ruta de *session.save_path* es mayor que 2, no se llevará a cabo la recolección de basura.

Aviso
Si lo deja apuntando a un directorio con permiso de lectura por el resto de usuarios, como `/tmp` (la opción por defecto), los demás usuarios del servidor pueden conseguir robar las sesiones obteniéndolas de la lista de archivos de ese directorio.

- *session.name* especifica el nombre de la sesión que se usa como nombre de la cookie. Sólo debería contener caracteres alfanuméricos. Por defecto vale *PHPSESSID*.
- *session.auto_start* especifica si el módulo de las sesión inicia una sesión automáticamente al comenzar la petición. Por defecto está *0* (desactivado).

- *session.cookie_lifetime* especifica la duración de la cookie en segundos que se manda al navegador. El valor *0* significa "hasta que se cierra el navegador", y es el que se encuentra por defecto.

- *session.serialize_handler* define el nombre del controlador que se utiliza para guardar y restaurar los datos. Actualmente se soportan un formato interno de PHP (cuyo nombre es *php*) y WDDX (cuyo nombre es *wddx*). WDDX sólo está disponible si PHP está compilado con Soporte para WDDX. Por defecto es *php*.

- *session.gc_probability* especifica la probabilidad de que se inicie la rutina gc (garbage collection - recoleción de basura) en cada petición en porcentaje. Por defecto es *1*.

- *session.gc_maxlifetime* especifica el número de segundos tras los cuales los datos se considerarán como 'basura' y serán eliminados.

- *session.referer_check* contiene la subcadena que usted quiera que se compruebe en cada "HTTP Referer" (N.T.: Página desde donde proviene el enlace a la página actual). Si el "Referer" fue enviado por el cliente y la subcadena no se ha encontrado, el session id incrustado será marcado como inválido. Por defecto es una cadena vacía.

- *session.entropy_file* indica la ruta a un recurso externo (un archivo) que se usará como fuente adicional de entropía en el proceso de creación de session id's. Por ejemplo */dev/random* o */dev/urandom*, que están disponibles en muchos sistemas Unix.

- *session.entropy_length* especifica el número de bytes que serán leidos del archivo indicado más arriba. Por defecto es *0* (desactivado).

- *session.use_cookies* indica si el módulo puede usar cookies para guardar el session id en el lado del cliente. Por defecto está a *1* (activado).

- *session.cookie_path* especifica la ruta a colocar en session_cookie. Por defecto es */*.

- *session.cookie_domain* especifica el dominio a establecer en session_cookie. Por defecto no se especifica ninguno en ningún caso.

- *session.cache_limiter* especifica el método de control del caché a usar en las páginas de la sesión (none/nocache/private/private_no_expire/public). Por defecto es *nocache* (no guardar las páginas en el caché).

- *session.cache_expire* especifica el tiempo-de-vida de las páginas de la sesión que se encuentran en el caché en minutos. No tiene efecto para el limitador nocache. Por defecto vale *180*.

- *session.use_trans_sid* indica si la inclusión del sid transparente está activada o no, si fue activada compilando con *--enable-trans-sid*. Por defecto está a *1* (activado).

- *url_rewriter.tags* especifica qué etiquetas html serán reescritas para incluir el session id si la inclusión del sid transparente está activada. Las etiquetas por defecto son *a=href,area=href,frame=src,input=src,form=fakeentry*

Nota: El manejo de sesiones fue añadido en PHP 4.0.

Tabla de contenidos

session_cache_limiter -- Lee y/o cambia el limitador del caché actual
session_commit -- Alias of **session_write_close()**
session_decode -- Decodifica los datos de una sesión a partir de una cadena
session_destroy -- Destruye todos los datos guardados en una sesión
session_encode -- Codifica los datos de la sesión actual en una cadena
session_get_cookie_params -- Obtiene los parámetros de la cookie de la sesión
session_id -- Lee y/o cambia el session id actual
session_is_registered -- Comprueba si una variable está registrada en la sesión
session_module_name -- Lee y/o cambia el módulo de la sesión actual
session_name -- Lee y/o cambia el nombre de la sesión actual
session_regenerate_id -- Actualizar la id de sesión actual con una recién generada
session_register -- Registrar una o más variables globales con la sesión actual
session_save_path -- Lee y/o cambia la ruta donde se guardan los datos de la sesión actual
session_set_cookie_params -- Cambia los parámetros de la cookie de la sesión
session_set_save_handler -- Establece unas funciones para el almacenamiento de los datos de la sesión a nivel de usuario
session_start -- Inicializar los datos de una sesión
session_unregister -- Desregistrar una variable de la sesión actual
session_unset -- Elimina todas las variables de la sesión
session_write_close -- Escribe los datos de la sesión y la finaliza

session_cache_expire

(PHP 4 >= 4.2.0, PHP 5)

session_cache_expire -- Devuelve la caducidad actual del caché

Descripción

int **session_cache_expire** ([int nueva_caducidad_cache])

session_cache_expire() devuelve la caducidad actual del caché. Si se proporciona *nueva_caducidad_cache*, se reemplazará la caducidad actual con *nueva_caducidad_cache*.

session_cache_limiter

(PHP 4 >= 4.0.3, PHP 5)

session_cache_limiter -- Lee y/o cambia el limitador del caché actual

Descripción

string **session_cache_limiter** ([string limitador_del_cache])

session_cache_limiter() devuelve el nombre del limitador de caché actual. Si se especifica *limitador_del_cache*, el nombre del limitador de caché actual se cambia al nuevo valor.

El limitador de caché controla las cabeceras HTTP de control del caché enviadas al cliente. Estas cabeceras determinan las reglas por las que el contenido de la página puede ser guardado en el caché local del cliente. Cambiando el limitador de caché a *nocache*, por ejemplo, impedirá cualquier tipo de almacenamiento en el caché por parte del cliente. Un valor de *public*, en cambio, permitiría el

almacenamiento en el caché. Tambié se puede cambiar a *private*, que es un poco más restrictivo que el *public*.

En el modo *private*, la cabecera Expire (caducidad) enviada al cliente puede confundir a algunos navegadores incluyendo Mozilla. Puede evitar este problema con el modo *private_no_expire*. La cabecera Expire nunca se envía al cliente en este modo.

> **Nota:** *private_no_expire* fue añadida en PHP 4.2.0dev.

Al comenzar la ejecución del script, el limitador de caché se reestablece al valor por defecto guardado en *session.cache_limiter*. De este modo, es necesario llamar a **session_cache_limiter()** en cada petición (y antes de llamar a **session_start()**).

Ejemplo 1. Ejemplos con session_cache_limiter()

```
<?php

# cambia el limitador del cach&eacute; a 'private'

session_cache_limiter('private');
$cache_limiter = session_cache_limiter();

echo "El limitador de cach&eacute; est&aacute; puesto ahora en $cache_limiter<p>";
?>
```

session_commit

session_commit -- Alias of **session_write_close()**

Description

This function is an alias of **session_write_close()**.

session_decode

(PHP 4 , PHP 5)

session_decode -- Decodifica los datos de una sesión a partir de una cadena

Descripción

bool **session_decode** (string datos)

session_decode() decodifica los datos de una sesión que se encuentran en *datos*, creando las variables guardadas en la sesión.

session_destroy

(PHP 4 , PHP 5)

session_destroy -- Destruye todos los datos guardados en una sesión

Descripción

bool **session_destroy** (void)

session_destroy() destruye todos los datos asociados con la sesión actual. No destruye ninguna de las variables globales asociadas a la sesión ni la cookie.

Esta función devuelve **TRUE** si se ha destruido la sesión correctamente y **FALSE** si ha habido algún problema al intentarlo.

Ejemplo 1. Destrucción de una sesión

```
<?php

// Inicializa de la sesi&oacute;n.
// Si est&aacute; usando session_name("algo"), &iexcl;no lo olvide ahora!
session_start();
// Destruye todas las variables de la sesi&oacute;n
session_unset();
// Finalmente, destruye la sesi&oacute;n
session_destroy();

?>
```

Ejemplo 2. Destrucción de una sesión con $_SESSION

```
<?php

// Inicializa la sesi&oacute;n.
// Si est&aacute; usando session_name("algo"), &iexcl;no lo olvide ahora!
session_start();
// Destruye todas las variables de la sesi&oacute;n
$_SESSION = array();
// Finalmente, destruye la sesi&oacute;n
session_destroy();

?>
```

session_encode

(PHP 4 , PHP 5)

session_encode -- Codifica los datos de la sesión actual en una cadena

Descripción

string **session_encode** (void)

session_encode() devuelve una cadena con el contenido de la sesión actual en su interior.

session_get_cookie_params

(PHP 4 , PHP 5)

session_get_cookie_params -- Obtiene los parámetros de la cookie de la sesión

Descripción

array **session_get_cookie_params** (void)

La función **session_get_cookie_params()** devuelve un vector con información sobre la cookie de la sesión actual, conteniendo los siguientes elementos:

- "lifetime" - La duración de la cookie.
- "path" - La ruta donde se guarda la información.
- "domain" - El dominio de la cookie.
- "secure" - La cookie debe ser enviada sólo bajo conexiones seguras. (Este elemento fue añadido en PHP 4.0.4.)

session_id

(PHP 4 , PHP 5)

session_id -- Lee y/o cambia el session id actual

Descripción

string **session_id** ([string id])

session_id() devuelve el session id de la sesión actual. Si se especifica un *id*, reemplazará el session id actual.

También se puede utilizar la constante SID para recuperar el nombre y el session id de la sesión actual como una cadena adecuada para añadir a las URLs.

session_is_registered

(PHP 4 , PHP 5)

session_is_registered -- Comprueba si una variable está registrada en la sesión

Descripción

bool **session_is_registered** (string nombre)

session_is_registered() devuelve **`TRUE`** si hay una variable registrada en la sesión actual cuyo nombre es *nombre*.

> **Nota:** Si utiliza *$_SESSION* (o *$HTTP_SESSION_VARS* con PHP 4.0.6 o inferior), use **isset()** para comprobar si una variable está registrada en *$_SESSION*.

Atención

Si utiliza *$HTTP_SESSION_VARS*/*$_SESSION*, no use **session_register()**, **session_is_registered()** ni **session_unregister()**.

session_module_name

(PHP 4 , PHP 5)

session_module_name -- Lee y/o cambia el módulo de la sesión actual

Descripción

string **session_module_name** ([string módulo])

session_module_name() devuelve el nombre del módulo de la sesión actual. Si se especifica *módulo*, se usará ese módulo en su lugar.

session_name

(PHP 4 , PHP 5)

session_name -- Lee y/o cambia el nombre de la sesión actual

Descripción

string **session_name** ([string nombre])

session_name() devuelve el nombre de la sesión actual. Si se especifica un *nombre*, el nombre de de la sesión actual se cambia a este valor.

El nombre de la sesión hace referencia al session id utilizado en las cookies y en las URLs. Debería contener únicamente caracteres alfanuméricos; también debería ser corto y descriptivo (p.ej. para usuarios con los avisos de las cookies activados). El nombre de la sesión se restaura al valor guardado por defecto en *session.name* al comenzar la petición. Así, pues, es necesario llamar a **session_name()** en cada petición (y antes de llamar a **session_start()** o a **session_register()**).

Ejemplo 1. Ejemplos de session_name()

```
<?php

// Cambiar el nombre de la sesi&oacute;n a WebsiteID

$nombre_anterior = session_name("WebsiteID");

echo "El anterior nombre de la sesi&oacute;n era $nombre_anterior<p>";
?>
```

session_regenerate_id

(PHP 4 >= 4.3.2, PHP 5)

session_regenerate_id -- Actualizar la id de sesión actual con una recién generada

Descripción

bool **session_regenerate_id** (void)

session_regenerate_id() reemplazará la id de sesión actual con una nueva, y conservará la información de sesión actual.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplo 1. Un ejemplo de session_regenerate_id()

```
<?php
session_start();

$id_sesion_antigua = session_id();

session_regenerate_id();

$id_sesion_nueva = session_id();

echo "Sesi&oacute;n Vieja: $id_sesion_antigua<br />";
echo "Sesi&oacute;n Nueva: $id_sesion_nueva<br />";

print_r($_SESSION);
?>
```

> **Nota:** A partir de PHP 4.3.3, si están habilitadas las cookies de sesión, el uso de **session_regenerate_id()** también enviará una nueva cookie de sesión con la nueva id de sesión.

Vea también **session_id()**, **session_start()**, y **session_name()**.

session_register

(PHP 4 , PHP 5)

session_register -- Registrar una o más variables globales con la sesión actual

Descripción

bool **session_register** (mixed nombre [, mixed ...])

session_register() acepta un número variable de argumentos, cualquiera de los cuales puede ser o una cadena que contiene el nombre de una variable, o una matriz que consista de nombres de variables u otras matrices. Para cada nombre, **session_register()** registra la variable global con ese nombre en la sesión actual.

Atención
Si desea que su script funcione independientemente de register_globals, necesita usar en su lugar la matriz $_SESSION, dado que las entradas de *$_SESSION* son registradas automáticamente. Si su script usa **session_register()**, no funcionará en entornos en donde la directiva PHP register_globals esté deshabilitada.

> **register_globals: Nota importante:** Desde PHP 4.2.0 el valor por defecto de la directiva register_globals es *off*. La comunidad PHP anima a todos a no confiar en esta directiva y usar en su lugar superglobals.

Atención
Esto registra una variable *global*. Si desea registrar una variable de sesión desde el interior de una función, necesita asegurarse de hacerla global usando la palabra clave **global** o la matriz *$GLOBALS[]*, o usar las matrices de sesión especiales, como se anota a continuación.

Atención
Si está usando *$_SESSION* (o *$HTTP_SESSION_VARS*), no use **session_register()**, **session_is_registered()**, ni **session_unregister()**.

Esta función devuelve **TRUE** cuando todas las variables son registradas satisfactoriamente en la sesión.

Si **session_start()** no fue llamada antes de que ésta función sea llamada, se realizará un llamado implícito a **session_start()** sin parámetro alguno. *$_SESSION* no imita este comportamiento y requiere **session_start()** antes de su uso.

Puede crear también una variable de sesión, simplemente definiendo el miembro apropiado de *$_SESSION* o la matriz *$HTTP_SESSION_VARS* (PHP < 4.1.0).

```
<?php
// El uso de session_register() es considerado obsoleto
$barney = "Un dinosaurio grande y violeta.";
session_register("barney");

// Se prefiere el uso de $_SESSION, a partir de PHP 4.1.0
$_SESSION["zim"] = "Un invasor de otro planeta.";

// El modo antiguo era usar $HTTP_SESSION_VARS
$HTTP_SESSION_VARS["bob_esponja"] = "&Eacute;l tiene pantalones cuadrados.";
?>
```

Nota: Actualmente es imposible registrar variables de recurso en una sesión. Por ejemplo, no puede crear una conexión a una base de datos y almacenar la id de conexión como una variable de sesión y esperar que la conexión aun sea válida la próxima vez que la sesión sea restaurada. Las funciones PHP que devuelven un recurso se identifican por tener un tipo de retorno de *resource* en su definicón de función. Una lista de funciones que devuelven recursos está disponible en el apéndice tipos de recurso.

Si *$_SESSION* (o *$HTTP_SESSION_VARS* para PHP 4.0.6 o versiones anteriores) es usado, asigne valores a *$_SESSION*. Por ejemplo: $_SESSION['var'] = 'ABC';

Vea también **session_is_registered()**, **session_unregister()**, y $_SESSION.

session_save_path

(PHP 4 , PHP 5)

session_save_path -- Lee y/o cambia la ruta donde se guardan los datos de la sesión actual

Descripción

string **session_save_path** ([string path])

session_save_path() devuelve la ruta del directorio usado actualmente para guardar los datos de la sesión. Si se especifica *path*, se cambiará la ruta donde se guardan los datos.

Nota: En algunos sistemas operativos, puede que quiera especificar una ruta en un sistema de archivos que maneja muchos archivos pequeños de forma eficiente. Por ejemplo, en Linux, reiserfs puede dar un rendimiento mejor que ext2fs.

session_set_cookie_params

(PHP 4 , PHP 5)

session_set_cookie_params -- Cambia los parámetros de la cookie de la sesión

Descripción

void **session_set_cookie_params** (int duración [, string path [, string dominio [, bool segura]]])

Cambia los parámetros de la cookie definidos en el archivo `php.ini`. El efecto de esta función sólo dura hasta que termina el script.

Nota: El parámetro *segura* fue añadido en PHP 4.0.4.

session_set_save_handler

(PHP 4 , PHP 5)

session_set_save_handler -- Establece unas funciones para el almacenamiento de los datos de la sesión a nivel de usuario

Descripción

bool **session_set_save_handler** (string abrir, string cerrar, string leer, string escribir, string destruir, string gc)

session_set_save_handler() establece las funciones que se utilizan a nivel de usuario para el almacenamiento y recuperación de los datos asociados a una sesión. Es lo más útil cuando se prefiere utilizar otro método de almacenamiento distinto del proporcionaddo por las sesiones de PHP. p.ej. Almacenar los datos de la sesión en una base de datos local. Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Nota: Debe cambiar la opción *session.save_handler* en la configuración a *user* en su archivo `php.ini` para que **session_set_save_handler()** tenga efecto.

Nota: El manejador "escribir" no se ejecuta hasta que se cierra la salida. Por ello, la salida de las sentencias que coloquemos en el manejador "escribir" para el depurado nunca será enviadas al navegador. Si se necesita producir una salida para el depurado, se sugiere que la salida se produzca en un archivo.

El siguiente ejemplo proporciona almacenamiento de las sesiones basado en archivos de forma similar al manejador de sesiones por defecto de PHP *files*. Este ejemplo puede ser extendido fácilmente para cubrir el almacenamiento en bases de datos usando su motor de soporte de bases de datos de PHP favorito.

La función de lectura debe devolver siempre una cadena para que el manejador de escritura funcione

como se espera. Devuelva una cadena vacía si no hay ningún dato a leer. Los valores devueltos de otros manejadores son convertidos a una expresión booleana. TRUE si todo ha ido correctamente, FALSE si ha habido algún problema.

Ejemplo 1. Ejemplo de session_set_save_handler()

```
<?php
function abrir ($save_path, $session_name) {
  global $sess_save_path, $sess_session_name;

  $sess_save_path = $save_path;
  $sess_session_name = $session_name;
  return(true);
}

function cerrar() {
  return(true);
}

function leer ($id) {
  global $sess_save_path, $sess_session_name;

  $sess_file = "$sess_save_path/sess_$id";
  if ($fp = @fopen($sess_file, "r")) {
    $sess_data = fread($fp, filesize($sess_file));
    return($sess_data);
  } else {
    return(""); // Debe devolver "" aqu&iacute;.
  }

}

function escribir ($id, $sess_data) {
  global $sess_save_path, $sess_session_name;

  $sess_file = "$sess_save_path/sess_$id";
  if ($fp = @fopen($sess_file, "w")) {
    return(fwrite($fp, $sess_data));
  } else {
    return(false);
  }

}

function destruir ($id) {
  global $sess_save_path, $sess_session_name;

  $sess_file = "$sess_save_path/sess_$id";
  return(@unlink($sess_file));
}

/*********************************************
 * ATENCI&Oacute;N - Necesitar&aacute; implementar alg&uacute;n     *
 * tipo de rutinas recolectoras de basura aqu&iacute; *
 *********************************************/
function rb ($maxlifetime) {
  return true;
}

session_set_save_handler ("abrir", "cerrar", "leer", "escribir", "destruir", "rb");

session_start();

// proceed to use sessions normally

?>
```

session_start

(PHP 4 , PHP 5)

session_start -- Inicializar los datos de una sesión

Descripción

bool **session_start** (void)

session_start() crea una sesió (o la continúa basandose en el session id pasado por GET o mediante una cookie).

Si desea usar una sesión con un nombre en concreto, debe llamar a **session_name()** antes de llamar a **session_start()**.

Esta función siempre devuelve **TRUE**.

> **Nota:** Si está usando sesiones basadas en las cookies, debe llamar a **session_start()** antes de que haya ninguna salida al navegador.

session_start() registrará un manejador de salida interno para la reescritura de las URL's si *trans-sid* está activado. Si un usuario utiliza *ob_gzhandler* o algo como **ob_start()**, el orden del manejador de salida es importante para que la salida sea la adecuada. Por ejemplo, el usuario debe registrar *ob_gzhandler* antes de iniciar la sesión.

> **Nota:** Se recomienda utilizar *zlib.output_compression* en lugar de *ob_gzhandler*

session_unregister

(PHP 4 , PHP 5)

session_unregister -- Desregistrar una variable de la sesión actual

Descripción

bool **session_unregister** (string nombre)

session_unregister() desregistra (olvida) la variable global llamada *nombre* de la sesión actual.

Esta función devuelve **TRUE** cuando la variable es eliminada de la sesión correctamente.

> **Nota:** Si utiliza *$_SESSION* (o *$HTTP_SESSION_VARS* con PHP 4.0.6 o inferior), use **unset()** para eliminar una variable de la sesión actual.

Atención
Esta función no borra la variable global correspondiente a *nombre*, sólo evita que la variable sea guardada como parte de la sesión. Debe llamar a **unset()** para eliminar la variable global correspondiente.

Atención
Si está trabajando con *$HTTP_SESSION_VARS*/*$_SESSION*, no utilice **session_register()**, **session_is_registered()** ni **session_unregister()**.

session_unset

(PHP 4 , PHP 5)

session_unset -- Elimina todas las variables de la sesión

Descripción

void **session_unset** (void)

La función **session_unset()** elimina y libera el espacio ocupado por todas las variables de la sesión actual registradas.

> **Nota:** Si utiliza *$_SESSION* (o *$HTTP_SESSION_VARS* con PHP 4.0.6 o inferior), use **unset()** en su lugar para desregistrar una variable de la sesión. p.ej. $_SESSION = array ();

session_write_close

(PHP 4 >= 4.0.4, PHP 5)

session_write_close -- Escribe los datos de la sesión y la finaliza

Descripción

void **session_write_close** (void)

Finaliza la sesión actual y guarda sus datos

Los datos de la sesió se suelen guardar tras finalizar la ejecución de su script sin necesidad de llamar a **session_write_close()**, pero como los datos de la sesió están bloqueados para evitar escrituras concurrentes, sólo un script puede trabajar con una sesión a la vez. Cuando se usan framesets junto con sesiones, podrá comprobar que los frames se cargan uno a uno debido a este bloqueo. Puede reducir el tiempo necesario para cargar los frames finalizando la sesión tan pronto como haya terminado los cambios a las variables de la sesión.

CXII. Funciones de Memoria Compartida

Introducción

Shmop es un conjunto de funciones que permiten a PHP leer, escribir, crear y borrar de forma sencilla segmentos de memoria compartida de tipo UNIX. Se debe tener en cuenta que las versiones de Windows anteriores a Windows 2000 no soportan el uso de memoria compartida.

> **Nota:** En PHP 4.0.3, el nombre de todas estas funciones estaba precedido por el prefijo

shm y actualmente lo están por el prefijo *shmop*.

Requirimientos

No se necesitan bibliotecas externas para construir esta extensión

---

Instalación

Para usar las funciones de memoria compartida, se debe añadir el parámetro *--enable-shmop* a las opciones de configuración de PHP.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

Constantes predefinidas

Esta extensión no tiene ninguna constante definida.

Ejemplos

Ejemplo 1. Resumen de las operaciones con Memoria Compartida

```php
<?php

// Creacion de un segmento de memoria compartida de 100 bytes y con un
// identificador igual a 0xff3
$shm_id = shmop_open(0xff3, "c", 0644, 100);
if(!$shm_id) {
    echo "No se pudo crear el segmento de memoria compartida\n";
}

// Obtencion del tama&ntilde;o del segmento de memoria compartida
$shm_size = shmop_size($shm_id);
echo "Segmento de memoria: se han reservado ".$shm_size. " bytes.\n";

// Escritura de una cadena de texto de prueba en la memoria compartida
$shm_bytes_written = shmop_write($shm_id, "mi segmento de memoria compartida",
0);
if($shm_bytes_written != strlen("mi segmento de memoria compartida")) {
    echo "No se pudo escribir todos los datos indicados\n";
}

// Lectura de la cadena de texto de prueba
$my_string = shmop_read($shm_id, 0, $shm_size);
if(!$my_string) {
    echo "No se pudo leer el segmento de memoria compartida\n";
}
echo "Los datos que contenia el segmento de memoria compartida son los
siguientes:".$my_string."\n";

// Borrado y eliminacion del segmento de memoria compartida
if(!shmop_delete($shm_id)) {
    echo "No se pudo borrar el segmento de memoria compartida.";
}
shmop_close($shm_id);

?>
```

Tabla de contenidos

shmop_close

(PHP 4 >= 4.0.4, PHP 5)

shmop_close -- Cierra un segmento de memoria compartida

Descripción

int **shmop_close** (int shmid)

shmop_close() se utiliza para cerrar un segmento de memoria compartida.

shmop_close() utiliza como parámetro el identificador del segmento de memoria compartida devuelto por la función **shmop_open()**.

Ejemplo 1. Cierre de un segmento de memoria compartida

```
<?php
shmop_close($shm_id);
?>
```

En el ejemplo anterior se cierra un segmento de memoria compartida cuyo identificador es*$shm_id*.

shmop_delete

(PHP 4 >= 4.0.4, PHP 5)

shmop_delete -- Borra un segmento de memoria compartida

Descripción

int **shmop_delete** (int shmid)

shmop_delete() se utiliza para borrar un segmento de memoria compartida.

shmop_delete() utiliza como parámetro el identificador del segmento de memoria compartida devuelto por la función **shmop_open()**. Si la función tiene éxito, devolverá un 1 (uno) y si no lo tiene devolverá un 0 (cero).

Ejemplo 1. Borrado de un segmento de memoria compartida

```
<?php
shmop_delete($shm_id);
?>
```

En el ejemplo anterior se borra un segmento de memoria compartida cuyo identificador es *$shm_id*.

shmop_open

(PHP 4 >= 4.0.4, PHP 5)

shmop_open -- Crea o abre un segmento de memoria compartida

Descripción

int **shmop_open** (int clave, string opciones, int modo, int tamanio)

shmop_open() se utiliza para crear o abrir un segmento de memoria compartida.

shmop_open() utiliza 4 parametros: clave, que es el identificador que el sistema utilizará para ese segmento de memoria compartida y que puede ser indicado tanto en formato decimal como hexadecimal. El segundo parámetro son las opciones que se pueden utilizar:

- "a" para acceder (se activa SHM_RDONLY en el shmat) se utiliza esta opción para abrir un segmento de memoria compartida que ya existe en modo de solo lectura.
- "c" para crear (se activa IPC_CREATE) se utiliza esta opción para crear un nuevo segmento de memoria compartida o para intentar abrir en modo de lectura y escritura un segmento que ya existe.
- "w" para acceder en modo de lectura y escritura. Se utiliza esta opción para poder leer un

segmento de memoria compartida y además poder escribir en el. Se trata de la opción más habitual en la mayoría de los casos.

- "n" para crear un nuevo segmento de memoria compartida (se activan IPC_CREATE| IPC_EXCL) se utiliza esta opción para crear un nuevo segmento de memoria compartida a no ser que ya exista otro segmento de memoria compartida con la misma clave. Esta opción es útil por motivos de seguridad para evitar por ejemplo "condiciones de carrera" (en inglés "race conditions").

El tercer parámetro es el modo, que son los permisos que se van a asignar al segmento de memoria compartida. Estos permisos son similares a los que se asignan a los archivos. La forma de indicar los permisos es identica a la que se utiliza en sistemas UNIX para indicar los permisos de los archivos, como por ejemplo "0644". El último parámetro es el tamaño en bytes del segmento de memoria compartida que se va a crear.

> **Nota:** Nota: el tercer y cuarto parámetro deben ser igual a 0 (cero) si se está abriendo un segmento de memoria compartida existente. Si la función **shmop_open()** tiene éxito, devolverá un identificador que se puede utilizar posteriormente para acceder al segmento de memoria compartida que se ha creado.

Ejemplo 1. Creación de un nuevo segmento de memoria compartida

```
<?php
$shm_key = ftok(__FILE__,'t');
$shm_id = shmop_open($shm_key, "c", 0644, 100);
?>
```

En el ejemplo anterior se crea un nuevo segmento de memoria compartida cuya clave se ha generado con ayuda de la función **ftok()**.

shmop_read

(PHP 4 >= 4.0.4, PHP 5)

shmop_read -- Lee un segmento de memoria compartida

Descripción

string **shmop_read** (int shmid, int comienzo, int posiciones)

shmop_read() se utiliza para leer datos almacenados en un segmento de memoria compartida.

shmop_read() utiliza 3 parámetros: shmid, que es el identificador del segmento de memoria compartida devuelto por la función **shmop_open()**; comienzo, que es la posición desde la que se empezarán a leer los datos del segmento de memoria y posiciones, que indica el número de bytes que se leerán a partir del comienzo indicado anteriormente.

Ejemplo 1. Lectura de un segmento de memoria compartida

```
<?php
$shm_data = shmop_read($shm_id, 0, 50);
?>
```

En el ejemplo anterior se leen los 50 primeros bytes del segmento de memoria y se almacenan los datos en la variable *$shm_data*.

shmop_size

(PHP 4 >= 4.0.4, PHP 5)

shmop_size -- Obtiene el tamaño de un segmento de memoria compartida

Descripción

int **shmop_size** (int shmid)

shmop_size() se utiliza para obtener el tamaño en bytes de un segmento de memoria compartida.

shmop_size() utiliza como parámetro el identificador del segmento de memoria compartida devuelto por la función **shmop_open()**. La función devuelve un dato de tipo int que contiene el tamaño en bytes del segmento de memoria compartida.

Ejemplo 1. Obtención del tamaño de un segmento de memoria compartida

```
<?php
$shm_size = shmop_size($shm_id);
?>
```

En el ejemplo anterior se almacena en la variable *$shm_size* el tamaño en bytes del segmento de memoria compartida cuyo identificador es *$shm_id*.

shmop_write

(PHP 4 >= 4.0.4, PHP 5)

shmop_write -- Escribe datos en un segmento de memoria compartida

Descriptión

int **shmop_write** (int shmid, string datos, int comienzo)

shmop_write() se utiliza para escribir datos en un segmento de memoria compartida.

shmop_write() utiliza 3 parámetros: shmid, que es el identificador del segmento de memoria compartida devuelto por la función **shmop_open()**; datos, que es una cadena que contiene los datos que se quieren escribir en el segmento de memoria compartida y comienzo, que indica la posición desde la que se empezarán a escribir los datos.

Ejemplo 1. Escritura en un segmento de memoria compartida

```
<?php
$shm_bytes_written = shmop_write($shm_id, $my_string, 0);
?>
```

En el anterior ejemplo se escriben en el segmento de memoria compartida los datos contenidos en la variable *$my_string*. La variable *$shm_bytes_written* contendrá el número de bytes escritos.

CXIII. SimpleXML functions

Introducción

The SimpleXML extension provides a very simple and easily usable toolset to convert XML to an object that can be processed with normal property selectors and array iterators.

Requirimientos

The SimpleXML extension requires PHP 5.

Instalación

The SimpleXML extension is enabled by default. To disable it, use the *--disable-simplexml* configure option.

Ejemplos

Many examples in this reference require an XML string. Instead of repeating this string in every example, we put it into a file which we include in each example. This included file is shown in the following example section. Alternatively, you could create an XML document and read it with **simplexml_load_file()**.

Ejemplo 1. Include file example.php with XML string

```
<?php
$xmlstr = <<<XML
<?xml version='1.0' standalone='yes'?>
<movies>
 <movie>
  <title>PHP: Behind the Parser</title>
  <characters>
   <character>
    <name>Ms. Coder</name>
    <actor>Onlivia Actora</actor>
   </character>
   <character>
    <name>Mr. Coder</name>
    <actor>El Act&#211;r</actor>
   </character>
  </characters>
  <plot>
   So, this language. It's like, a programming language. Or is it a
   scripting language? All is revealed in this thrilling horror spoof
   of a documentary.
  </plot>
  <rating type="thumbs">7</rating>
  <rating type="stars">5</rating>
 </movie>
</movies>
XML;
?>
```

The simplicity of SimpleXML appears most clearly when one extracts a string or number from a

basic XML document.

Ejemplo 2. Getting *<plot>*

```
<?php
include 'example.php';

$xml = simplexml_load_string($xmlstr);

echo $xml->movie[0]->plot; // "So this language. It's like..."
?>
```

Ejemplo 3. Accessing non-unique elements in SimpleXML

When multiple instances of an element exist as children of a single parent element, normal iteration techniques apply.

```
<?php
include 'example.php';

$xml = simplexml_load_string($xmlstr);

/* For each <movie> node, we echo a separate <plot>. */
foreach ($xml->movie as $movie) {
   echo $movie->plot, '<br />';
}

?>
```

Ejemplo 4. Using attributes

So far, we have only covered the work of reading element names and their values. SimpleXML can also access element attributes. Access attributes of an element just as you would elements of an **array**.

```
<?php
include 'example.php';

$xml = simplexml_load_string($xmlstr);

/* Access the <rating> nodes of the first movie.
 * Output the rating scale, too. */
foreach ($xml->movie[0]->rating as $rating) {
    switch((string) $rating['type']) { // Get attributes as element indices
    case 'thumbs':
        echo $rating, ' thumbs up';
        break;
    case 'stars':
        echo $rating, ' stars';
        break;
    }
}
?>
```

Ejemplo 5. Comparing Elements and Attributes with Text

To compare an element or attribute with a string or pass it into a function that requires a string, you must cast it to a string using *(string)*. Otherwise, PHP treats the element as an object.

```
<?php
include 'example.php';

$xml = simplexml_load_string($xmlstr);

if ((string) $xml->movie->title == 'PHP: Behind the Parser') {
    print 'My favorite movie.';
}

htmlentities((string) $xml->movie->title);
?>
```

Ejemplo 6. Using Xpath

SimpleXML includes builtin Xpath support. To find all <*character*> elements:

```
<?php
include 'example.php';
$xml = simplexml_load_string($xmlstr);

foreach ($xml->xpath('//character') as $character) {
    echo $character->name, 'played by ', $character->actor, '<br />';
}
?>
```

'//' serves as a wildcard. To specify absolute paths, omit one of the slashes.

Ejemplo 7. Setting values

Data in SimpleXML doesn't have to be constant. The object allows for manipulation of all of its elements.

```
<?php
include 'example.php';
$xml = simplexml_load_string($xmlstr);

$xml->movie[0]->characters->character[0]->name = 'Miss Coder';

echo $xml->asXML();
?>
```

The above code will output a new XML document, just like the original, except that the new XML will change Ms. Coder to Miss Coder.

Ejemplo 8. DOM Interoperability

PHP has a mechanism to convert XML nodes between SimpleXML and DOM formats. This example shows how one might change a DOM element to SimpleXML.

```
<?php
$dom = new domDocument;
$dom->loadXML('<books><book><title>blah</title></book></books>');
if (!$dom) {
     echo 'Error while parsing the document';
     exit;
}

$s = simplexml_import_dom($dom);

echo $s->book[0]->title;
?>
```

Tabla de contenidos

SimpleXMLElement->asXML

(no version information, might be only in CVS)

SimpleXMLElement->asXML -- Return a well-formed XML string based on SimpleXML element

Description

string **SimpleXMLElement->asXML** (void)

The *asXML* method formats the parent object's data in XML version 1.0.

Ejemplo 1. Get XML

```
<?php
$string = <<<XML
<a>
 <b>
  <c>text</c>
  <c>stuff</c>
 </b>
 <d>
  <c>code</c>
 </d>
</a>
XML;

$xml = simplexml_load_string($string);

echo $xml->asXML(); // <?xml ... <a><b><c>text</c><c>stuff</c> ...

?>
```

asXML also works on Xpath results:

Ejemplo 2. Using asXML() on Xpath results

```
<?php
// Continued from example XML above.

/* Search for <a><b><c> */
$result = $xml->xpath('/a/b/c');

while(list( , $node) = each($result)) {
    echo $node->asXML(); // <c>text</c> and <c>stuff</c>
}
?>
```

SimpleXMLElement->attributes

(no version information, might be only in CVS)

SimpleXMLElement->attributes -- Identifies an element's attributes

Description

SimpleXMLElement **simplexml_element->attributes** (string data)

This function provides the attributes and values defined within an xml tag.

> **Nota:** SimpleXML ha creado una regla de añadir propiedades iterativas a la mayoria de metodos. No se pueden ver usando **var_dump()** o cualquier cosa que examine objetos.

Ejemplo 1. Interpret an XML string

```
<?php
$string = <<<XML
<a>
 <foo name="one" game="lonely">1</foo>
</a>
XML;

$xml = simplexml_load_string($string);
foreach($xml->foo[0]->attributes() as $a => $b) {
    echo $a,'="',$b,"\"\n";
}
?>
```

This script will display:

```
name="one"
game="lonely"
```

SimpleXMLElement->children

(no version information, might be only in CVS)

SimpleXMLElement->children -- Finds children of given node

Description

SimpleXMLElement **simplexml_element->children** (void)

This method finds the children of the element of which it is a member. The result follows normal iteration rules.

> **Nota:** SimpleXML ha creado una regla de añadir propiedades iterativas a la mayoria de metodos. No se pueden ver usando **var_dump()** o cualquier cosa que examine objetos.

Ejemplo 1. Traversing a *children()* pseudo-array

```
<?php
$xml = simplexml_load_string(
'<person>
 <child role="son">
  <child role="daughter"/>
 </child>
 <child role="daughter">
  <child role="son">
   <child role="son"/>
  </child>
 </child>
</person>');

foreach ($xml->children() as $second_gen) {
    echo ' The person begot a ' . $second_gen['role'];

    foreach ($second_gen->children() as $third_gen) {
        echo ' who begot a ' . $third_gen['role'] . ';';

        foreach ($third_gen->children() as $fourth_gen) {
            echo ' and that ' . $third_gen['role'] .
                ' begot a ' . $fourth_gen['role'];
        }
    }
}
?>
```

This script will output:

```
The person begot a son who begot a daughter; The person
begot a daughter who begot a son; and that son begot a son
```

SimpleXMLElement->xpath

(no version information, might be only in CVS)

SimpleXMLElement->xpath -- Runs Xpath query on XML data

Description

array **SimpleXMLElement->xpath** (string path)

The *xpath* method searches the SimpleXML node for children matching the Xpath *path*. It always returns an **array** of SimpleXMLElement objects.

Ejemplo 1. Xpath

```
<?php
$string = <<<XML
<a>
 <b>
  <c>text</c>
  <c>stuff</c>
 </b>
 <d>
  <c>code</c>
 </d>
</a>
XML;

$xml = simplexml_load_string($string);

/* Search for <a><b><c> */
$result = $xml->xpath('/a/b/c');

while(list( , $node) = each($result)) {
    echo '/a/b/c: ',$node,"\n";
}

/* Relative paths also work... */
$result = $xml->xpath('b/c');

while(list( , $node) = each($result)) {
    echo 'b/c: ',$node,"\n";
}
?>
```

This script will display:

```
/a/b/c: text
/a/b/c: stuff
b/c: text
b/c: stuff
```

Notice that the two results are equal.

simplexml_import_dom

(PHP 5)

simplexml_import_dom -- Get a *SimpleXMLElement* object from a DOM node.

Description

SimpleXMLElement **simplexml_import_dom** (DOMNode node [, string class_name])

This function takes a node of a DOM document and makes it into a SimpleXML node. This new object can then be used as a native SimpleXML element. If any errors occur, it returns **FALSE**.

Ejemplo 1. Import DOM

```
<?php
$dom = new domDocument;
$dom->loadXML('<books><book><title>blah</title></book></books>');
if (!$dom) {
    echo 'Error while parsing the document';
    exit;
}

$s = simplexml_import_dom($dom);

echo $s->book[0]->title; // blah
?>
```

See also **dom_import_simplexml()**.

simplexml_load_file

(PHP 5)

simplexml_load_file -- Interprets an XML file into an object

Description

object **simplexml_load_file** (string filename [, string class_name [, int options]])

This function will convert the well-formed XML document in the file specified by *filename* to an **object** of class SimpleXMLElement. If any errors occur during file access or interpretation, the function returns **FALSE**.

You may use the optional *class_name* parameter so that **simplexml_load_file()** will return an object of the specified class. That class should extend the SimpleXMLElement class.

Since PHP 5.1.0 and Libxml 2.6.0, you may also use the *options* parameter to specify additional Libxml parameters.

Ejemplo 1. Interpret an XML document

```
<?php
// The file test.xml contains an XML document with a root element
// and at least an element /[root]/title.

if (file_exists('test.xml')) {
    $xml = simplexml_load_file('test.xml');

    var_dump($xml);
} else {
    exit('Failed to open test.xml.');
}
?>
```

This script will display, on success:

```
SimpleXMLElement Object
(
  [title] => Example Title
  ...
)
```

At this point, you can go about using *$xml->title* and any other elements.
See also: **simplexml_load_string()**

simplexml_load_string

(PHP 5)

simplexml_load_string -- Interprets a string of XML into an object

Description

object **simplexml_load_string** (string data [, string class_name [, int options]])

This function will take the well-formed xml string *data* and return an **object** of class SimpleXMLElement with properties containing the data held within the xml document. If any errors occur, it returns **FALSE**.

You may use the optional *class_name* parameter so that **simplexml_load_string()** will return an object of the specified class. That class should extend the SimpleXMLElement class.

Since PHP 5.1.0 and Libxml 2.6.0, you may also use the *options* parameter to specify additional Libxml parameters.

Ejemplo 1. Interpret an XML string

```
<?php
$string = <<<XML
<?xml version='1.0'?>
<document>
 <title>Forty What?</title>
 <from>Joe</from>
 <to>Jane</to>
 <body>
  I know that's the answer -- but what's the question?
 </body>
</document>
XML;

$xml = simplexml_load_string($string);

var_dump($xml);
?>
```

This script will display:

```
SimpleXMLElement Object
(
  [title] => Forty What?
  [from] => Joe
  [to] => Jane
  [body] =>
   I know that's the answer -- but what's the question?
)
```

At this point, you can go about using *$xml->body* and such.
See also: **simplexml_load_file()**.

CXIV. Funciones SNMP

Para usar las funciones SNMP en Unix necesita instalar el paquete UCD SNMP. En Windows estas funciones están solamente disponibles en NT y no en Win95/98.

Importante: Para usar el paquete UCD SNMP, necesita definir NO_ZEROLENGTH_COMMUNITY a 1 antes de compilarlo. Despues de configurar UCD SNMP, edite config.h y busque NO_ZEROLENGTH_COMMUNITY. Descomente la línea #define. Debería de verse como sigue:

```
#define NO_ZEROLENGTH_COMMUNITY 1
```

Si ve faltas de segmentación desconocidas en combinación con los comandos SNMP, no siga las siguientes instrucciones. Si no desea recompilar UCD SNMP, puede compilar PHP con la opción --enable-ucd-snmp-hack la cual trabajará entorno a las mismas características.

Tabla de contenidos

snmp_get_quick_print

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

snmp_get_quick_print -- Va a buscar el valor actual de la biblioteca UCD estableciendo quick_print

Descripción

boolean **snmp_get_quick_print** (void)

Delvuele el valor actual almacenado en la biblioteca UCD para quick_print. quick_print está desactivado por defecto.

```
$quickprint = snmp_get_quick_print();
```

La llamada a la función superior podría devolver **FALSE** si quick_print está activo, y **TRUE** si quick_print está activo.

snmp_get_quick_print() está solamente disponible cuando estemos usando la biblioteca UCD SNMP. Esta función no está disponible cuando estemos usando la biblioteca Windows SNMP.

Ver: **snmp_get_quick_print()** para una descripción completa de lo que hace quick_print.

snmp_get_valueretrieval

(PHP 4 >= 4.3.3, PHP 5)

snmp_get_valueretrieval -- Return the method how the SNMP values will be returned

Description

int **snmp_get_valueretrieval** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

snmp_read_mib

(PHP 5)

snmp_read_mib -- Reads and parses a MIB file into the active MIB tree

Description

int **snmp_read_mib** (string filename)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

snmp_set_enum_print

(PHP 4 >= 4.3.0, PHP 5)

snmp_set_enum_print -- Return all values that are enums with their enum value instead of the raw integer

Description

void **snmp_set_enum_print** (int enum_print)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

snmp_set_oid_numeric_print

(PHP 4 >= 4.3.0, PHP 5)

snmp_set_oid_numeric_print -- Return all objects including their respective object id within the specified one

Description

void **snmp_set_oid_numeric_print** (int oid_numeric_print)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

snmp_set_quick_print

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

snmp_set_quick_print -- Establece el valor de quick_print con el de la biblioteca UCD SNMP.

Descripción

void **snmp_set_quick_print** (boolean quick_print)

Establece el valor de quick_print con la biblioteca UCD SNMP. Cuando esto está establecido (1), la biblioteca SNMP devolverá valores 'quick printed'. De esta manera sólo el valor será impreso. Cuando quick_print no está activada (por defecto) la biblioteca UCD SNMP imprime información extra incluyendo el tipo del valor (p. Ej. IPAddress o OID). Adicionalmente, si quick_print no está activado, la biblioteca imprime valores hexadecimales adicionales para todas las cadenas de 3 o menos caracteres.

El ajuste de quick_print es generalmente usado cuando usando la información devuelta con anterioridad se muestra.

```
snmp_set_quick_print(0);
$a = snmpget("127.0.0.1", "public", ".1.3.6.1.2.1.2.2.1.9.1");
echo "$a<BR>\n";
snmp_set_quick_print(1);
$a = snmpget("127.0.0.1", "public", ".1.3.6.1.2.1.2.2.1.9.1");
echo "$a<BR>\n";
```

El primer valor impreso debe de ser: 'Timeticks: (0) 0:00:00.00', donde qick_print se activa, solo se imprimira '0:00:00.00'.

Por defecto la biblioteca UCD SNMP devuelve valores detallados, quick_print es usado para devolver solamente el valor.

Las cadenas son mantenidas normalmente con comillas extra, esto será corregido en versiones posteriores.

snmp_get_quick_print() está sólo disponible cuando estemos usando la biblioteca UCD SNMP.

Esta función no está disponible cuando estemos usando la biblioteca Windows SNMP.

snmp_set_valueretrieval

(PHP 4 >= 4.3.3, PHP 5)

snmp_set_valueretrieval -- Specify the method how the SNMP values will be returned

Description

int **snmp_set_valueretrieval** (int method)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

snmpget

(PHP 3, PHP 4 , PHP 5)

snmpget -- Va a buscar un objeto SNMP

Descripción

string **snmpget** (string hostname, string community, string object_id [, int timeout [, int retries]])

Devuelve el valor de un objeto SNMP en caso de exito y **`FALSE`** en caso de error.

La función **snmpget()** es usada para leer el valor de un objeto SNMP especificado por el *object_id*. El agente SNMP es especificado por el *hostname* y la comunidad lectora es especificada por el parametro *community*.

$syscontact = snmpget("127.0.0.1", "public", "system.SysContact.0")

snmpgetnext

(PHP 5)

snmpgetnext -- Fetch a SNMP object

Description

string **snmpgetnext** (string host, string community, string object_id [, int timeout [, int retries]])

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

snmprealwalk

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

snmprealwalk -- Return all objects including their respective object ID within the specified one

Description

array **snmprealwalk** (string host, string community, string object_id [, int timeout [, int retries]])

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

snmpset

(PHP 3>= 3.0.12, PHP 4 , PHP 5)

snmpset -- Va a buscar un objeto SNMP

Descripción

string **snmpset** (string hostname, string community, string object_id, string type, mixed value [, int timeout [, int retries]])

Establece el valor especificado para el objeto SNMP, devolviendo **`TRUE`** en caso de exito o **`FALSE`** en caso de error.

La función **snmpset()** es usada para establecer el valor de un objeto SNMP especificado por el *object_id*. El agente SNMP es espcificado por el *hostname* y la comunidad lectora por el parametro *community*.

snmpwalk

(PHP 3, PHP 4 , PHP 5)

snmpwalk -- Busqueda por un arbol de información acerca de un entidad de red

Descripción

array **snmpwalk** (string hostname, string community, string object_id [, int timeout [, int retries]])

Devuelve una matriz de valores de objetos SMNP comenzando por el **object_id()** como raíz y **`FALSE`** en caso de error.

La función **snmpwalk()** es usada para leer todos los valores de un agente SNMP especificado por el *hostname*. *Community* especifica la comunidad lectora para el agente. Un *object_id* nulo se toma como la raíz del arbol de los objetos SNMP y todos los objetos por debajo de ese arbol son devueltos como una matriz. Si *object_id* es especificado, todos los objetos SNMP por debajo de *object_id* son devueltos.

```
$a = snmpwalk("127.0.0.1", "public", "");
```

Encima de una función de llamada podrían devolverse todos los objetos SNMP del agente SNMP en ejecución en el servidor local. Uno puede pasar por todos los valores con un bucle.

```
for ($i=0; $i<count($a); $i++) {
    echo $a[$i];
}
```

snmpwalkoid

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

snmpwalkoid -- Busqueda por un arbol de información acerca de un entidad de red

Descripción

array **snmpwalkoid** (string hostname, string community, string object_id [, int timeout [, int retries]])

Devuelve una matriz asociativa con los identificadores de los objetos y sus respectivos valores comenzando por el *object_id* como raíz y **FALSE** en caso de error.

La función **snmpwalkoid()** es usada para leer todos los identificadores de objetos y sus respectivos valores de un agente SNMP especificado por el nombre del servidor. La lectura de *community* especifica la comunidad para el agente. Un *object_id* nulo es tomado como la raíz del arbol de objetos SNMP y todos los objetos por debajo de este arbol son devueltos como una matriz. Si *object_id* es especificado, todos los objetos SNMP inferiores al *object_id* son devueltos.

La existencia de **snmpwalkoid()** y **snmpwalk()** tiene razones historicas. Ambas funciones son proporcionadas para compatibilidad hacia atrás.

```
$a = snmpwalkoid("127.0.0.1", "public", "");
```

La llamada a las funciones superiores devuelve todos los objetos SNMP del agente SNMP en ejecución en el servidor local. Uno puede pasar por todos los valores con un bucle.

```
for (reset($a); $i = key($a); next($a)) {
    echo "$i: $a[$i]<br>\n";
}
```

CXV. SOAP Functions

Introducción

The SOAP extension can be used to write SOAP Servers and Clients. It supports subsets of SOAP

1.1, SOAP 1.2 and WSDL 1.1 specifications.

Requirimientos

This extension makes use of the GNOME xml library. Download and install this library. You will need at least libxml-2.5.4.

Instalación

This extension is only available if PHP was configured with *--enable-soap*.

Configuración en tiempo de ejecución

El comportamiento de estas funciones está afectado por los valores definidos en `php.ini`.

Tabla 1. SOAP Configuration Options

Name	Default	Changeable
soap.wsdl_cache_enabled	"1"	PHP_INI_ALL
soap.wsdl_cache_dir	"/tmp"	PHP_INI_ALL
soap.wsdl_cache_ttl	86400	PHP_INI_ALL

For further details and definitions of the PHP_INI_* constants, see the Apéndice H.

A continuación se presenta una corta explicación de las directivas de configuración.

soap.wsdl_cache_enabled **boolean**

Enables or disables the WSDL caching feature.

soap.wsdl_cache_dir **string**

Sets the directory name where the SOAP extension will put cache files.

soap.wsdl_cache_ttl **int**

Sets the number of seconds (time to live) that cached files will be used instead the originals.

Clases predefinidas

SoapClient

Constructor

- SoapClient->__construct() - constructs a new SoapClient object

---

Métodos

- SoapClient->__call() - Calls a SOAP function (deprecated)
- SoapClient->__doRequest() - Performs a SOAP request
- SoapClient->__getFunctions() - Returns list of SOAP functions
- SoapClient->__getLastRequest() - Returns last SOAP request
- SoapClient->__getLastRequestHeaders() - Returns last SOAP request headers
- SoapClient->__getLastResponse() - Returns last SOAP response
- SoapClient->__getLastResponseHeaders() - Returns last SOAP response headers
- SoapClient->__getTypes() - Returns list of SOAP types
- SoapClient->__setCookie() - Sets the cookie that will be sent with the SOAP request
- SoapClient->__soapCall() - Calls a SOAP function

SoapFault

Constructor

- SoapFault->__construct() - construct a new SoapFault object

---

SoapHeader

SoapHeader is a special low-level class for passing or returning SOAP headers. It's just a data holder and it does not have any special methods except its constructor. It can be used in the SoapClient->__soapCall() method to pass a SOAP header or in a SOAP header handler to return the header in a SOAP response.

Constructor

- SoapHeader->__construct() - construct a new SoapHeader object

---

SoapParam

SoapParam is a special low-level class for naming parameters and returning values in *non-WSDL* mode. It's just a data holder and it does not have any special methods except its constructor.

Constructor

- SoapParam->__construct() - construct a new SoapParam object

---

SoapServer

Constructor

- SoapServer->__construct() - construct a new SoapServer object

---

Métodos

- SoapServer->addFunction() - Adds one or several functions those will handle SOAP requests
- SoapServer->fault() -
- SoapServer->getFunctions() - Returns list of defined functions
- SoapServer->handle() - Handles a SOAP request
- SoapServer->setClass() - Sets class which will handle SOAP requests
- SoapServer->setPersistence() - Sets persistence mode of SoapServer

---

SoapVar

SoapVar is a special low-level class for encoding parameters and returning values in *non-WSDL* mode. It's just a data holder and does not have any special methods except the constructor. It's useful when you want to set the type property in SOAP request or response.

Constructor

- SoapVar->__construct() - construct a new SoapVar object

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

SOAP_1_1 (integer)

SOAP_1_2 (integer)

SOAP_PERSISTENCE_SESSION (integer)

SOAP_PERSISTENCE_REQUEST (integer)

SOAP_FUNCTIONS_ALL (integer)

SOAP_ENCODED (integer)

SOAP_LITERAL (integer)

SOAP_RPC (integer)

SOAP_DOCUMENT (integer)

SOAP_ACTOR_NEXT (integer)

SOAP_ACTOR_NONE (integer)

SOAP_ACTOR_UNLIMATERECEIVER (integer)

SOAP_COMPRESSION_ACCEPT (integer)

SOAP_COMPRESSION_GZIP (integer)

SOAP_COMPRESSION_DEFLATE (integer)

UNKNOWN_TYPE (integer)

XSD_STRING (integer)

XSD_BOOLEAN (integer)

XSD_DECIMAL (integer)

XSD_FLOAT (integer)

XSD_DOUBLE (integer)

XSD_DURATION (integer)

XSD_DATETIME (integer)

XSD_TIME (integer)

XSD_DATE (integer)

XSD_GYEARMONTH (integer)

XSD_GYEAR (integer)

XSD_GMONTHDAY (integer)

XSD_GDAY (integer)

XSD_GMONTH (integer)

XSD_HEXBINARY (integer)

XSD_BASE64BINARY (integer)

XSD_ANYURI (integer)

XSD_QNAME (integer)

XSD_NOTATION (integer)

XSD_NORMALIZEDSTRING (integer)

XSD_TOKEN (integer)

XSD_LANGUAGE (integer)

XSD_NMTOKEN (integer)

XSD_NAME (integer)

XSD_NCNAME (integer)

XSD_ID (integer)

XSD_IDREF (integer)

XSD_IDREFS (integer)

XSD_ENTITY (integer)

XSD_ENTITIES (integer)

XSD_INTEGER (integer)

XSD_NONPOSITIVEINTEGER (integer)

XSD_NEGATIVEINTEGER (integer)

XSD_LONG (integer)

XSD_INT (integer)

XSD_SHORT (integer)

XSD_BYTE (integer)

XSD_NONNEGATIVEINTEGER (integer)

XSD_UNSIGNEDLONG (integer)

XSD_UNSIGNEDINT (integer)

XSD_UNSIGNEDSHORT (integer)

XSD_UNSIGNEDBYTE (integer)

XSD_POSITIVEINTEGER (integer)

XSD_NMTOKENS (integer)

XSD_ANYTYPE (integer)

SOAP_ENC_OBJECT (integer)

SOAP_ENC_ARRAY (integer)

XSD_1999_TIMEINSTANT (integer)

XSD_NAMESPACE (string)

XSD_1999_NAMESPACE (string)

Tabla de contenidos

SoapServer->getFunctions() -- Returns list of defined functions
SoapServer->handle() -- Handles a SOAP request
SoapServer->setClass() -- Sets class which will handle SOAP requests
SoapServer->setPersistence() -- Sets persistence mode of SoapServer
SoapVar->__construct() -- SoapVar constructor
use_soap_error_handler() --

is_soap_fault

(PHP 5)

is_soap_fault -- Checks if SOAP call was failed

Descripción

bool **is_soap_fault** (mixed obj)

This function is useful when you like to check if the SOAP call failed, but don't like to use exceptions. To use it you must create a **SoapClient** object with the *exceptions* option set to zero or **FALSE**. In this case, the SOAP method will return a special **SoapFault** object which encapsulates the fault details (faultcode, faultstring, faultactor and faultdetails).

If *exceptions* is not set then SOAP call will throw an exception on error. **is_soap_fault()** checks if the given parameter is a **SoapFault** object.

Lista de parámetros

obj

The tested object.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplos

Ejemplo 1. is_soap_fault() example

```
<?php
$client = new SoapClient("some.wsdl", array('exceptions' => 0));
$result = $client->SomeFunction();
if (is_soap_fault($result)) {
    trigger_error("SOAP Fault: (faultcode: {$result->faultcode}, faultstring: {$result-
}
?>
```

Ejemplo 2. SOAP's standard method for error reporting is exceptions

```
<?php
try {
    $client = new SoapClient("some.wsdl");
    $result = $client->SomeFunction(/* ... */);
} catch (SoapFault $fault) {
    trigger_error("SOAP Fault: (faultcode: {$fault->faultcode}, faultstring: {$fault->f
}
?>
```

Ver también

SoapClient->__construct()
SoapFault->__construct()

SoapClient->__call()

SoapClient->__call() -- Calls a SOAP function (deprecated)

Descripción

class **SoapClient** {

mixed **__call** (string function_name [, array arguments [, array options [, array input_headers [, array output_headers]]]])

}

This method is deprecated. Use SoapClient->__soapCall() instead of it.

SoapClient->__construct()

SoapClient->__construct() -- SoapClient constructor

Descripción

class **SoapClient** {

__construct (mixed wsdl [, array options])

}

This constructor creates **SoapClient** objects in *WSDL* or *non-WSDL* mode.

Lista de parámetros

wsdl

URI of the *WSDL* file or **NULL** if working in *non-WSDL* mode.

options

An array of options. If working in WSDL mode, this parameter is optional. If working in non-WSDL mode, you must set the *location* and *uri* options, where *location* is the URL to request and *uri* is the target namespace of the SOAP service.

The *style* and *use* options only work in non-WSDL mode. In WSDL mode, they come from the WSDL file.

The *soap_version* option specifies whether to use SOAP 1.1, or SOAP 1.2 client.

For HTTP authentication, you may use the *login* and *password* options. For making an HTTP connection through a proxy server, use the options *proxy_host*, *proxy_port*, *proxy_login* and *proxy_password*. For HTTPS client certificate authentication use *local_cert* and *passphrase* options.

The *compression* option allows to use compression of HTTP SOAP requests and responses.

The *encoding* option defines internal character encoding. This option does not change the encofing of SOAP requests (it is always utf-8), but converts strings into it.

The *classmap* option can be used to map some WSDL types to PHP classes. This option must be an array with WSDL types as keys and names of PHP classes as values.

The *trace* and *exceptions* options are useful for debuging purpose.

Ejemplos

Ejemplo 1. SoapClient examples

```
<?php

$client = new SoapClient("some.wsdl");

$client = new SoapClient("some.wsdl", array('soap_version'   => SOAP_1_2));

$client = new SoapClient("some.wsdl", array('login'          => "some_name",
                                            'password'       => "some_password"));

$client = new SoapClient("some.wsdl", array('proxy_host'     => "localhost",
                                            'proxy_port'     => 8080));

$client = new SoapClient("some.wsdl", array('proxy_host'     => "localhost",
                                            'proxy_port'     => 8080,
                                            'proxy_login'    => "some_name",
                                            'proxy_password' => "some_password"));

$client = new SoapClient("some.wsdl", array('local_cert'     => "cert_key.pem"));

$client = new SoapClient(null, array('location' => "http://localhost/soap.php",
                                     'uri'      => "http://test-uri/"));

$client = new SoapClient(null, array('location' => "http://localhost/soap.php",
                                     'uri'      => "http://test-uri/",
                                     'style'    => SOAP_DOCUMENT,
                                     'use'      => SOAP_LITERAL));

$client = new SoapClient("some.wsdl",
  array('compression' => SOAP_COMPRESSION_ACCEPT | SOAP_COMPRESSION_GZIP));

$server = new SoapClient("some.wsdl", array('encoding'=>'ISO-8859-1'));

class MyBook {
        public $title;
        public $author;
}

$server = new SoapClient("books.wsdl", array('classmap' => array('book' => "MyBook")));

?>
```

SoapClient->__doRequest()

SoapClient->__doRequest() -- Performs a SOAP request

Descripción

class **SoapClient** {

string **__doRequest** (string request, string location, string action, int version)

}

Performs SOAP request over HTTP.

This method can be overridden in subclasses to implement different transport layers, perform additional XML processing or other purpose.

Lista de parámetros

request

The XML SOAP request.

location

The URL to request.

action

The SOAP action.

action

The SOAP version.

Valores retornados

The XML SOAP response.

Ejemplos

Ejemplo 1. Some examples

```
<?php
function Add($x,$y) {
  return $x+$y;
}

class LocalSoapClient extends SoapClient {

  function __construct($wsdl, $options) {
    parent::__construct($wsdl, $options);
    $this->server = new SoapServer($wsdl, $options);
    $this->server->addFunction('Add');
  }

  function __doRequest($request, $location, $action, $version) {
    ob_start();
    $this->server->handle($request);
    $response = ob_get_contents();
    ob_end_clean();
    return $response;
  }

}

$x = new LocalSoapClient(NULL,array('location'=>'test://',
                                    'uri'=>'http://testuri.org'));
var_dump($x->Add(3,4));
?>
```

SoapClient->__getFunctions()

SoapClient->__getFunctions() -- Returns list of SOAP functions

Descripción

class **SoapClient** {

array **__getFunctions** (void)

}

Returns the list of SOAP functions.

> **Nota:** This function works only in WSDL mode.

Valores retornados

The list of SOAP functions.

Ejemplos

Ejemplo 1. SoapClient->__getFunctions() example

```
<?php
$client = SoapClient("some.wsdl");
var_dump($client->__getFunctions());
?>
```

Ver también

SoapClient->__construct()

SoapClient->__getLastRequest()

SoapClient->__getLastRequest() -- Returns last SOAP request

Descripción

class **SoapClient** {

string **__getLastRequest** (void)

}

> **Nota:** This method works only if the **SoapClient** object was created with the *trace* option.

Valores retornados

The last SOAP request.

Ejemplos

Ejemplo 1. SoapClient->__getLastRequest() example

```
<?php
$client = SoapClient("some.wsdl", array('trace' => 1));
$result = $client->SomeFunction();
echo "REQUEST:\n" . $client->__getLastRequest() . "\n";
?>
```

Ver también

SoapClient->__construct()
SoapClient->__getLastRequestHeaders()
SoapClient->__getLastResponse()
SoapClient->__getLastResponseHeaders()

SoapClient->__getLastRequestHeaders()

SoapClient->__getLastRequestHeaders() -- Returns last SOAP request headers

Descripción

class **SoapClient** {

string **__getLastRequestHeaders** (void)

}

> **Nota:** This method works only if the **SoapClient** object was created with the *trace* option.

Valores retornados

The last SOAP request headers.

Ver también

SoapClient->__construct()
SoapClient->__getLastRequest()
SoapClient->__getLastResponse()
SoapClient->__getLastResponseHeaders()

SoapClient->__getLastResponse()

SoapClient->__getLastResponse() -- Returns last SOAP response.

Descripción

class **SoapClient** {

string **__getLastResponse** (void)

}

> **Nota:** This method works only if the **SoapClient** object was created with the *trace* option.

Valores retornados

The last SOAP response.

Ejemplos

Ejemplo 1. SoapClient->__getLastResponse() example

```
<?php
$client = SoapClient("some.wsdl", array('trace' => 1));
$result = $client->SomeFunction();
echo "RESPONSE:\n" . $client->__getLastResponse() . "\n";
?>
```

Ver también

SoapClient->__construct()
SoapClient->__getLastResponseHeaders()
SoapClient->__getLastRequest()
SoapClient->__getLastRequestHeaders()

SoapClient->__getLastResponseHeaders()

SoapClient->__getLastResponseHeaders() -- Returns last SOAP response headers.

Descripción

class **SoapClient** {

string **__getLastResponseHeaders** (void)

}

> **Nota:** This method works only if the **SoapClient** object was created with the *trace* option.

Valores retornados

The last SOAP response headers.

Ver también

SoapClient->__construct()
SoapClient->__getLastResponse()
SoapClient->__getLastRequest()
SoapClient->__getLastRequestHeaders()

SoapClient->__getTypes()

SoapClient->__getTypes() -- Returns list of SOAP types

Descripción

class **SoapClient** {

array **__getTypes** (void)

}

This function works only in WSDL mode.

Valores retornados

The list of SOAP types.

Ejemplos

Ejemplo 1. SoapClient->__getTypes() example

```
<?php
$client = new SoapClient("some.wsdl");
var_dump($client->__getTypes());
?>
```

Ver también

SoapClient->__construct()

SoapClient->__setCookie()

SoapClient->__setCookie() -- Sets the cookie that will be sent with the SOAP request

Descripción

class **SoapClient** {

void **__setCookie** (string name [, string value])

}

Defines a cookie to be sent along with the SOAP requests.

> **Nota:** Calling this method will affect all following calls to **SoapClient** methods.

Lista de parámetros

name

The name of the cookie.

value

The value of the cookie. If not specified, the cookie will be deleted.

Valores retornados

No value is returned.

SoapClient->__soapCall()

SoapClient->__soapCall() -- Calls a SOAP function

Descripción

class **SoapClient** {

mixed **__soapCall** (string function_name [, array arguments [, array options [, array input_headers [, array output_headers]]]])

}

This is a low level API function that is used to make a SOAP call. Usually, in WSDL mode, you can simply call SOAP functions as **SoapClient** methods. This method useful in non-WSDL mode when *soapaction* is unknown, *uri* differs from the default or when sending and/or receiving SOAP Headers.

On error, a call to a SOAP function can cause PHP to throw exceptions or return a **SoapFault** object if exceptions are disabled. To check if the function call failed to catch the SoapFault exceptions, check the result with **is_soap_fault()**.

Valores retornados

SOAP functions may return one, or multiple values. If only one value is returned by the SOAP function, the return value of *__soapCall* will be a simple value (e.g. an integer, a string, etc). If multiple values are returned, *__soapCall* will return an associative array of named output parameters.

Ejemplos

Ejemplo 1. SoapClient->__soapCall() Examples

```
<?php

$client = new SoapClient("some.wsdl");
$client->SomeFunction($a, $b, $c);

$client->__soapCall("SomeFunction", array($a, $b, $c));
$client->__soapCall("SomeFunction", array($a, $b, $c), NULL,
                    new SoapHeader(), $output_headers);


$client = new SoapClient(null, array('location' => "http://localhost/soap.php",
                                     'uri'      => "http://test-uri/"));
$client->SomeFunction($a, $b, $c);
$client->__soapCall("SomeFunction", array($a, $b, $c));
$client->__soapCall("SomeFunction", array($a, $b, $c),
                    array('soapaction' => 'some_action',
                          'uri'        => 'some_uri'));
?>
```

Ver también

SoapClient->__construct()
SoapParam->__construct()
SoapVar->__construct()
SoapHeader->__construct()
SoapFault->__construct()
is_soap_fault()

SoapFault->__construct()

SoapFault->__construct() -- SoapFault constructor

Descripción

class **SoapFault** {

__construct (string faultcode, string faultstring [, string faultactor [, mixed detail [, string faultname [, SoapHeader headerfault]]]])

}

This class is useful when you would like to send SOAP fault responses from the PHP handler. *faultcode*, *faultstring*, *faultactor* and *details* are standard elements of SOAP Fault;

Lista de parámetros

faultcode

The error code of the **SoapFault**.

faultstring

The error message of the **SoapFault**.

faultactor

A string identifying the actor that caused the error.

detail

faultname

Can be used to select the proper fault encoding from WSDL.

headerfault

Can be used during SOAP header handling to report an error in the response header.

Ejemplos

Ejemplo 1. Some examples

```
<?php
function test($x)
{
    return new SoapFault("Server", "Some error message");
}

$server = new SoapServer(null, array('uri' => "http://test-uri/"));
$server->addFunction("test");
$server->handle();
?>
```

It is possible to use PHP exception mechanism to throw SOAP Fault.

Ejemplo 2. Some examples

```
<?php
function test($x)
{
    throw new SoapFault("Server", "Some error message");
}

$server = new SoapServer(null, array('uri' => "http://test-uri/"));
$server->addFunction("test");
$server->handle();
?>
```

Ver también

SoapClient->__construct()
SoapClient->__soapCall()
SoapVar->__construct()
SoapParam->__construct()
SoapFault->__construct()
is_soap_fault()

SoapHeader->__construct()

SoapHeader->__construct() -- SoapHeader constructor

Descripción

class **SoapHeader** {

__construct (string namespace, string name [, mixed data [, bool mustUnderstand [, mixed actor]]])

}

Constructs a new **SoapHeader** object.

Lista de parámetros

namespace

The namespace of the SOAP header element.

name

The name of the SOAP header element.

data

A SOAP header's content. It can be a PHP value or a **SoapVar** object.

mustUnderstand

Value of the *mustUnderstand* attribute of the SOAP header element.

actor

Value of the *actor* attribute of the SOAP header element.

Ejemplos

Ejemplo 1. Some examples

```
<?php
$client = new SoapClient(null, array('location' => "http://localhost/soap.php",
                                     'uri'      => "http://test-uri/"));
$client->__call("echoVoid", null, null,
                new SoapHeader('http://soapinterop.org/echoheader/',
                               'echoMeStringRequest',
                               'hello world'));
?>
```

Ver también

SoapClient->__soapCall()
SoapVar->__construct()
SoapParam->__construct()

SoapParam->__construct()

SoapParam->__construct() -- SoapParam constructor

Descripción

class **SoapParam** {

__construct (mixed data, string name)

}

Constructs a new **SoapParam** object.

Lista de parámetros

data

The data to pass or return. You can pass this parameter directly as PHP value, but in this case it will be named as *paramN* and the SOAP Service may not understand it.

name

The parameter name.

Ejemplos

Ejemplo 1. Some examples

```
<?php
$client = new SoapClient(null,array('location' => "http://localhost/soap.php",
                                    'uri'      => "http://test-uri/"));
$client->SomeFunction(new SoapParam($a, "a"),
                      new SoapParam($b, "b"),
                      new SoapParam($c, "c"));
?>
```

Ver también

SoapClient->__soapCall()
SoapVar->__construct()

SoapServer->addFunction()

SoapServer->addFunction() -- Adds one or several functions those will handle SOAP requests

Descripción

class **SoapServer** {

void **addFunction** (mixed functions)

}

Exports one or more functions for remote clients.

Lista de parámetros

functions

To export one function, pass the function name into this parameter as a string.

To export several functions, pass an array of function names.

To export all the functions, pass a special constant **SOAP_FUNCTIONS_ALL**.

Nota: *functions* must receive all input arguments in the same order as defined in the WSDL file (They should not receive any output parameters as arguments) and return one or more values. To return several values they must return an array with named output parameters.

Valores retornados

No value is returned.

Ejemplos

Ejemplo 1. Some examples

```
<?php

function echoString($inputString)
{
    return $inputString;
}

$server->addFunction("echoString");

function echoTwoStrings($inputString1, $inputString2)
{
    return array("outputString1" => $inputString1,
                 "outputString2" => $inputString2);
}
$server->addFunction(array("echoString", "echoTwoStrings"));

$server->addFunction(SOAP_FUNCTIONS_ALL);

?>
```

Ver también

SoapServer->__construct()
SoapServer->setClass()

SoapServer->__construct()

SoapServer->__construct() -- SoapServer constructor

Descripción

class **SoapServer** {

__construct (mixed wsdl [, array options])

}

This constructor allows the creation of **SoapServer** objects in WSDL or non-WSDL mode.

Lista de parámetros

wsdl

> If you want the WSDL mode, you must set this to the URI of a WSDL file. In the other case, you must set this to **NULL** and set the *uri* option.

options

> Allow setting a default SOAP version (*soap_version*), internal character encoding (*encoding*), and actor URI (*actor*). The *classmap* option can be used to map some WSDL types to PHP classes. This option must be an array with WSDL types as keys and names of PHP classes as values.

Ejemplos

Ejemplo 1. Some examples

```
<?php

$server = new SoapServer("some.wsdl");

$server = new SoapServer("some.wsdl", array('soap_version' => SOAP_1_2));

$server = new SoapServer("some.wsdl", array('actor' => "http://example.org/ts-tests/C"));

$server = new SoapServer("some.wsdl", array('encoding'=>'ISO-8859-1'));

$server = new SoapServer(null, array('uri' => "http://test-uri/"));

class MyBook {
        public $title;
        public $author;
}

$server = new SoapServer("books.wsdl", array('classmap' => array('book' => "MyBook")));

?>
```

SoapServer->fault()

SoapServer->fault() --

Descripción

class **SoapServer** {

void **fault** (string code, string string [, string actor [, mixed details [, string name]]])

}

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

Ver también

SoapFault->__construct()

SoapServer->getFunctions()

SoapServer->getFunctions() -- Returns list of defined functions

Descripción

class **SoapServer** {

array **getFunctions** (void)

}

This method returns the list of all functions added by SoapServer->addFunction() or SoapServer->setClass().

Valores retornados

The list of all functions.

Ejemplos

Ejemplo 1. Some examples

```
<?php
$server = new SoapServer(NULL, array("uri" => "http://test-uri"));
$server->addFunction(SOAP_FUNCTIONS_ALL);
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $server->handle();
} else {
  echo "This SOAP server can handle following functions: ";
  $functions = $server->getFunctions();
  foreach($functions as $func) {
    echo $func . "\n";
  }
}
?>
```

Ver también

SoapServer->__construct()

SoapServer->addFunction()
SoapServer->setClass()

SoapServer->handle()

SoapServer->handle() -- Handles a SOAP request

Descripción

class **SoapServer** {

void **handle** ([string soap_request])

}

Processes a SOAP request, calls necessary functions, and sends a response back.

Lista de parámetros

soap_request

> The SOAP request. If this argument is omitted, the request is supposed to be in the *$HTTP_RAW_POST_DATA* PHP variable.

Valores retornados

No value is returned.

Ejemplos

Ejemplo 1. Some examples

```
<?php
function test($x)
{
    return $x;
}

$server = new SoapServer(null, array('uri' => "http://test-uri/"));
$server->addFunction("test");
$server->handle();
?>
```

Ver también

SoapServer->__construct()

SoapServer->setClass()

SoapServer->setClass() -- Sets class which will handle SOAP requests

Descripción

class **SoapServer** {

void **setClass** (string class_name [, mixed args])

}

Exports all methods from specified class.

The object can be made persistent across request for a given PHP session with the SoapServer->setPersistence() method.

Lista de parámetros

class_name

The name of the exported class.

args

This optional parameter will be passed to the default class constructor during object creation.

Valores retornados

No value is returned.

Ejemplos

Ejemplo 1. Some examples

```
<?php

class foo {
    function foo()
    {
    }
}
$server->setClass("foo");

class bar {
    function bar($x, $y)
    {
    }
}
$server->setClass("bar", $arg1, $arg2);

?>
```

Ver también

SoapServer->__construct()
SoapServer->addFunction()
SoapServer->setPersistence()

SoapServer->setPersistence()

SoapServer->setPersistence() -- Sets persistence mode of SoapServer

Descripción

class **SoapServer** {

void **setPersistence** (int mode)

}

This function allows saving data between requests in a PHP session. It works only with a server that exports functions from a class with SoapServer->setClass().

Lista de parámetros

mode

One of the *SOAP_PERSISTENCE_XXX* constants.

Valores retornados

No value is returned.

Ejemplos

Ejemplo 1. Some examples

```
<?php

$server->setPersistence(SOAP_PERSISTENCE_SESSION);

$server->setPersistence(SOAP_PERSISTENCE_REQUEST);

?>
```

Ver también

SoapServer->__construct()
SoapServer->setClass()

SoapVar->__construct()

SoapVar->__construct() -- SoapVar constructor

Descripción

class **SoapVar** {

__construct (mixed data, int encoding [, string type_name [, string type_namespace [, string

node_name [, string node_namespace]]]])

}

Constructs a new **SoapVar** object.

Lista de parámetros

data

The data to pass or return.

encoding

The encoding ID, one of the *XSD_...* constants.

type_name

The type name.

type_namespace

The type namespace.

node_name

The XML node name.

node_namespace

The XML node namespace.

Ejemplos

Ejemplo 1. Some examples

```
<?php
class SOAPStruct {
    function SOAPStruct($s, $i, $f)
    {
        $this->varString = $s;
        $this->varInt = $i;
        $this->varFloat = $f;
    }
}
$client = new SoapClient(null, array('location' => "http://localhost/soap.php",
                                     'uri'      => "http://test-uri/"));
$struct = new SOAPStruct('arg', 34, 325.325);
$soapstruct = new SoapVar($struct, SOAP_ENC_OBJECT, "SOAPStruct", "http://soapinterop.c
$client->echoStruct(new SoapParam($soapstruct, "inputStruct"));
?>
```

Ver también

SoapClient->__soapCall()
SoapParam->__construct()

use_soap_error_handler()

(no version information, might be only in CVS)

use_soap_error_handler() --

Descripción

void **use_soap_error_handler** ([bool handler])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

CXVI. Funciones de Socket

Introducción

La extensión de sockets implementa una interfaz de bajo nivel con las funciones de comunicación de sockets, basadas en los populares sockets BSD, ofreciendo la posibilidad de actuar como un servidor de sockets, así como cliente.

Para una interfaz más genérica de sockets del lado del cliene, vea **stream_socket_client()**, **stream_socket_server()**, **fsockopen()**, y **pfsockopen()**.

Cuando use estas funciones, es importante recordar que, aunque muchas de ellas tienen nombres idénticos a sus contrapartes en C, usualmente cuentan con declaraciones distintas. Por favor, asegúrese de leer las descripciones, con el propósito de evitar confusiones.

Aquellos que no se encuentran familiarizados con programación de sockets, pueden encontrar una gran cantidad de material útil en las páginas man de Unix apropiadas, y existe una enorme cantidad de información estilo tutorial sobre programación de sockets en C en la web, mucha de la cual puede aplicarse, con ligeras modificaciones, a la programación de sockets en PHP. El FAQ de Sockets Unix puede ser un buen comienzo.

Aviso
Esta extensión es *EXPERIMENTAL*. Esto significa que el comportamiento de esta extensión, los nombre de sus funciones y en definitiva TODO lo documentado sobre esta extensión, puede cambiar en una futura versión de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Requirimientos

No se necesitan bibliotecas externas para construir esta extensión

---

Instalación

The socket functions described here are part of an extension to PHP which must be enabled at compile time by giving the *--enable-sockets* option to **configure**.

> **Nota:** El soporte para IPv6 fue agregado en *PHP 5.0.0* .

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

`AF_UNIX` (integer)

`AF_INET` (integer)

`AF_INET6` (integer)

`SOCK_STREAM` (integer)

`SOCK_DGRAM` (integer)

`SOCK_RAW` (integer)

`SOCK_SEQPACKET` (integer)

`SOCK_RDM` (integer)

`MSG_OOB` (integer)

`MSG_WAITALL` (integer)

`MSG_PEEK` (integer)

`MSG_DONTROUTE` (integer)

`SO_DEBUG` (integer)

SO_REUSEADDR (integer)

SO_KEEPALIVE (integer)

SO_DONTROUTE (integer)

SO_LINGER (integer)

SO_BROADCAST (integer)

SO_OOBINLINE (integer)

SO_SNDBUF (integer)

SO_RCVBUF (integer)

SO_SNDLOWAT (integer)

SO_RCVLOWAT (integer)

SO_SNDTIMEO (integer)

SO_RCVTIMEO (integer)

SO_TYPE (integer)

SO_ERROR (integer)

SOL_SOCKET (integer)

PHP_NORMAL_READ (integer)

PHP_BINARY_READ (integer)

SOL_TCP (integer)

SOL_UDP (integer)

Errores de Socket

La extensión de sockets fue escrita para ofrecer una interfaz usable a los poderosos sockets BSD. Se ha tenido cuidado para que las funciones trabajen igualmente bien en implementaciones Win32 y Unix. Casi todas las funciones de sockets pueden fallar bajo ciertas condiciones y por lo tanto emiten un mensaje **E_WARNING** que describe el error. Algunas veces esto no ocurre conforme al deseo del desarrollador. Por ejemplo, la función **socket_read()** puede emitir repentinamente un mensaje **E_WARNING** debido a que la conexión ha sido cerrada inesperadamente. Es común suprimir la advertencia con el operador @ y atrapar el código de error dentro de la aplicación con la función **socket_last_error()**. Puede llamar la función **socket_strerror()** con éste código de error para recuperar una cadena que describe el error. Vea su descripción para más información.

Nota: Los mensajes **E_WARNING** generados por la extensión de sockets se encuentran en Inglés, aunque el mensaje de error recuperado aparecerá dependiendo de la localidad actual (**LC_MESSAGES**):

```
Warning - socket_bind() unable to bind address [98]: Die Adresse wird bereits verw
```

Ejemplos

Ejemplo 1. Ejemplo de Socket: Servidor TCP/IP simple

Este ejemplo le muestra un servidor simple que repite de vuelta su entrada. Modifique las variables *direccion* *puerto* para que se acomoden a su configuración y ejecútelo. Puede entonces conectarse con el servidor media un comando similar a: **telnet 192.168.1.53 10000** (en donde la dirección y el puerto se ajustan a su configurac Cualquier cosa que escriba será entonces impresa en el lado del servidor, y devuelta a su lado. Para desconect ingrese 'salir'.

```
#!/usr/local/bin/php -q
<?php
error_reporting(E_ALL);

/* Permitir que el script permanezca en espera de conexiones. */
set_time_limit(0);

/* Habilitar vaciado de salida implicito, de modo que veamos lo que
 * obtenemos a medida que va llegando. */
ob_implicit_flush();

$direccion = '192.168.1.53';
$puerto    = 10000;

if (($sock = socket_create(AF_INET, SOCK_STREAM, SOL_TCP)) < 0) {
    echo "socket_create() fall&oacute;: motivo: " . socket_strerror($sock) . "\n";
}

if (($ret = socket_bind($sock, $direccion, $puerto)) < 0) {
    echo "socket_bind() fall&oacute;: motivo: " . socket_strerror($ret) . "\n";
}

if (($ret = socket_listen($sock, 5)) < 0) {
    echo "socket_listen() fall&oacute;: motivo: " . socket_strerror($ret) . "\n";
}

do {
    if (($mens_sock = socket_accept($sock)) < 0) {
        echo "socket_accept() fall&oacute;: motivo " . socket_strerror($mens_sock) . "\
        break;
    }
    /* Enviar instrucciones. */
    $mensaje = "\nBienvenido al Servidor de Prueba PHP. \n" .
               "Para salir, escriba 'salir'. " .
               "Para detener el servidor, escriba 'detener'.\n";
    socket_write($mens_sock, $mensaje, strlen($mensaje));

    do {
        if (false === ($buf = socket_read($mens_sock, 2048, PHP_NORMAL_READ))) {
            echo "socket_read() fall&oacute;: motivo: " . socket_strerror($ret) . "\n";
            break 2;
        }
        if (!$buf = trim($buf)) {
            continue;
        }
        if ($buf == 'salir') {
            break;
        }
        if ($buf == 'detener') {
            socket_close($mens_sock);
            break 2;
        }
        $respuesta = "PHP: Usted dijo '$buf'.\n";
        socket_write($mens_sock, $respuesta, strlen($respuesta));
        echo "$buf\n";
    } while (true);
    socket_close($mens_sock);
} while (true);

socket_close($sock);
?>
```

Ejemplo 2. Ejemplo de Socket: Cliente TCP/IP simple

Este ejemplo muestra un cliente HTTP simple, de un paso. Simplemente se conecta con una página, envia una petición HEAD, imprime la respuesta, y sale.

```php
<?php
error_reporting(E_ALL);

echo "<h2>Conexi&oacute;n TCP/IP</h2>\n";

/* Obtener el puerto para el servicio WWW. */
$puerto_servicio = getservbyname('www', 'tcp');

/* Obtener la direccion IP del host de destino. */
$direccion = gethostbyname('www.example.com');

/* Crear un socket TCP/IP. */
$socket = socket_create(AF_INET, SOCK_STREAM, SOL_TCP);
if ($socket < 0) {
    echo "socket_create() fall&oacute;: motivo: " . socket_strerror($socket) . "\n";
} else {
    echo "OK.\n";
}

echo "Intentando una conexi&oacute;n con '$direccion' en el puerto '$puerto_servicio'..
$resultado = socket_connect($socket, $direccion, $puerto_servicio);
if ($resultado < 0) {
    echo "socket_connect() fall&oacute;.\nMotivo: ($resultado) " .
         socket_strerror($resultado) . "\n";
} else {
    echo "OK.\n";
}

$entrada = "HEAD / HTTP/1.1\r\n";
$entrada .= "Host: www.example.com\r\n";
$entrada .= "Connection: Close\r\n\r\n";
$salida = '';

echo "Enviando petici&oacute;n HTTP HEAD...";
socket_write($socket, $entrada, strlen($entrada));
echo "OK.\n";

echo "Leyendo respuesta:\n\n";
while ($salida = socket_read($socket, 2048)) {
    echo $salida;
}

echo "Cerrando socket...";
socket_close($socket);
echo "OK.\n\n";
?>
```

Tabla de contenidos

socket_select -- Runs the select() system call on the given arrays of sockets with a specified timeout
socket_send -- Sends data to a connected socket
socket_sendto -- Sends a message to a socket, whether it is connected or not
socket_set_block -- Sets blocking mode on a socket resource
socket_set_nonblock -- Sets nonblocking mode for file descriptor fd
socket_set_option -- Sets socket options for the socket
socket_shutdown -- Shuts down a socket for receiving, sending, or both
socket_strerror -- Return a string describing a socket error
socket_write -- Write to a socket

socket_accept

(PHP 4 >= 4.1.0, PHP 5)

socket_accept -- Accepts a connection on a socket

Description

resource **socket_accept** (resource socket)

After the socket *socket* has been created using **socket_create()**, bound to a name with **socket_bind()**, and told to listen for connections with **socket_listen()**, this function will accept incoming connections on that socket. Once a successful connection is made, a new socket resource is returned, which may be used for communication. If there are multiple connections queued on the socket, the first will be used. If there are no pending connections, **socket_accept()** will block until a connection becomes present. If *socket* has been made non-blocking using **socket_set_blocking()** or **socket_set_nonblock()**, **FALSE** will be returned.

The socket resource returned by **socket_accept()** may not be used to accept new connections. The original listening socket *socket*, however, remains open and may be reused.

Returns a new socket resource on success, or **FALSE** on error. The actual error code can be retrieved by calling **socket_last_error()**. This error code may be passed to **socket_strerror()** to get a textual explanation of the error.

See also **socket_bind()**, **socket_connect()**, **socket_listen()**, **socket_create()**, and **socket_strerror()**.

socket_bind

(PHP 4 >= 4.1.0, PHP 5)

socket_bind -- Binds a name to a socket

Description

bool **socket_bind** (resource socket, string address [, int port])

socket_bind() binds the name given in *address* to the socket described by *socket*, which must be a valid socket resource created with **socket_create()**.

The *address* parameter is either an IP address in dotted-quad notation (e.g. *127.0.0.1*), if the socket

is of the **AF_INET** family; or the pathname of a Unix-domain socket, if the socket family is **AF_UNIX**.

The *port* parameter is only used when connecting to an **AF_INET** socket, and designates the port on the remote host to which a connection should be made.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo. The error code can be retrieved with **socket_last_error()**. This code may be passed to **socket_strerror()** to get a textual explanation of the error. Note that **socket_last_error()** is reported to return an invalid error code in case you are trying to bind the socket to a wrong address that does not belong to your Windows 9x/ME machine.

See also **socket_connect()**, **socket_listen()**, **socket_create()**, **socket_last_error()** and **socket_strerror()**.

socket_clear_error

(PHP 4 >= 4.2.0, PHP 5)

socket_clear_error -- Clears the error on the socket or the last error code

Description

void **socket_clear_error** ([resource socket])

This function clears the error code on the given socket or the global last socket error.

This function allows explicitly resetting the error code value either of a socket or of the extension global last error code. This may be useful to detect within a part of the application if an error occurred or not.

See also **socket_last_error()** and **socket_strerror()**.

socket_close

(PHP 4 >= 4.1.0, PHP 5)

socket_close -- Closes a socket resource

Description

void **socket_close** (resource socket)

socket_close() closes the socket resource given by *socket*.

> **Nota: socket_close()** can't be used on PHP file resources created with **fopen()**, **popen()**, **fsockopen()**, or **pfsockopen()**; it is meant for sockets created with **socket_create()** or **socket_accept()**.

See also **socket_bind()**, **socket_listen()**, **socket_create()** and **socket_strerror()**.

socket_connect

(PHP 4 >= 4.1.0, PHP 5)

socket_connect -- Initiates a connection on a socket

Description

bool **socket_connect** (resource socket, string address [, int port])

Initiates a connection using the socket resource *socket*, which must be a valid socket resource created with **socket_create()**.

The *address* parameter is either an IP address in dotted-quad notation (e.g. *127.0.0.1*), if the socket is of the **AF_INET** family; or the pathname of a Unix domain socket, if the socket family is **AF_UNIX**.

The *port* parameter is only used when connecting to an **AF_INET** socket, and designates the port on the remote host to which a connection should be made.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo. The error code can be retrieved with **socket_last_error()**. This code may be passed to **socket_strerror()** to get a textual explanation of the error.

See also **socket_bind()**, **socket_listen()**, **socket_create()**, **socket_last_error()** and **socket_strerror()**.

socket_create_listen

(PHP 4 >= 4.1.0, PHP 5)

socket_create_listen -- Opens a socket on port to accept connections

Description

resource **socket_create_listen** (int port [, int backlog])

This function is meant to ease the task of creating a new socket which only listens to accept new connections.

socket_create_listen() creates a new socket resource of type **AF_INET** listening on *all* local interfaces on the given port waiting for new connections.

The *backlog* parameter defines the maximum length the queue of pending connections may grow to. **SOMAXCONN** may be passed as *backlog* parameter, see **socket_listen()** for more information.

socket_create_listen() returns a new socket resource on success or **FALSE** on error. The error code can be retrieved with **socket_last_error()**. This code may be passed to **socket_strerror()** to get a textual explanation of the error.

> **Nota:** If you want to create a socket which only listens on a certain interface you need to

use **socket_create()**, **socket_bind()** and **socket_listen()**.

See also **socket_create()**, **socket_bind()**, **socket_listen()**, **socket_last_error()** and **socket_strerror()**.

socket_create_pair

(PHP 4 >= 4.1.0, PHP 5)

socket_create_pair -- Creates a pair of indistinguishable sockets and stores them in an array

Description

bool **socket_create_pair** (int domain, int type, int protocol, array &fd)

socket_create_pair() creates two connected and indistinguishable sockets, and stores them in *fd*. This function is commonly used in IPC (InterProcess Communication).

The *domain* parameter specifies the protocol family to be used by the socket.

Tabla 1. Available address/protocol families

Domain	Description
AF_INET	IPv4 Internet based protocols. TCP and UDP are common protocols of this protocol family. Supported only in windows.
AF_INET6	IPv6 Internet based protocols. TCP and UDP are common protocols of this protocol family. Support added in PHP 5.0.0. Supported only in windows.
AF_UNIX	Local communication protocol family. High efficiency and low overhead make it a great form of IPC (Interprocess Communication).

The *type* parameter selects the type of communication to be used by the socket.

Tabla 2. Available socket types

Type	Description
SOCK_STREAM	Provides sequenced, reliable, full-duplex, connection-based byte streams. An out-of-band data transmission mechanism may be supported. The TCP protocol is based on this socket type.
SOCK_DGRAM	Supports datagrams (connectionless, unreliable messages of a fixed maximum length). The UDP protocol is based on this socket type.
SOCK_SEQPACKET	Provides a sequenced, reliable, two-way connection-based data transmission path for datagrams of fixed maximum length; a consumer is required to read an entire packet with each read call.
SOCK_RAW	Provides raw network protocol access. This special type of socket can be used to manually construct any type of protocol. A common use for this socket type is to perform ICMP requests (like ping, traceroute, etc).
SOCK_RDM	Provides a reliable datagram layer that does not guarantee ordering. This is most likely not implemented on your operating system.

The *protocol* parameter sets the specific protocol within the specified *domain* to be used when

communicating on the returned socket. The proper value can be retrieved by name by using **getprotobyname()**. If the desired protocol is TCP, or UDP the corresponding constants **SOL_TCP**, and **SOL_UDP** can also be used.

Tabla 3. Common protocols

Name	Description
icmp	The Internet Control Message Protocol is used primarily by gateways and hosts to report errors in datagram communication. The "ping" command (present in most modern operating systems) is an example application of ICMP.
udp	The User Datagram Protocol is a connectionless, unreliable, protocol with fixed record lengths. Due to these aspects, UDP requires a minimum amount of protocol overhead.
tcp	The Transmission Control Protocol is a reliable, connection based, stream oriented, full duplex protocol. TCP guarantees that all data packets will be received in the order in which they were sent. If any packet is somehow lost during communication, TCP will automatically retransmit the packet until the destination host acknowledges that packet. For reliability and performance reasons, the TCP implementation itself decides the appropriate octet boundaries of the underlying datagram communication layer. Therefore, TCP applications must allow for the possibility of partial record transmission.

Ejemplo 1. socket_create_pair() example

```
<?php
$sockets = array();
$uniqid = uniqid('');
if (file_exists("/tmp/$uniqid.sock")) {
    die('Temporary socket already exists.');
}
/* Setup socket pair */
if (!socket_create_pair(AF_UNIX, SOCK_STREAM, 0, $sockets)) {
    echo socket_strerror(socket_last_error());
}
/* Send and Recieve Data */
if (!socket_write($sockets[0], "ABCdef123\n", strlen("ABCdef123\n"))) {
    echo socket_strerror(socket_last_error());
}
if (!$data = socket_read($sockets[1], strlen("ABCdef123\n"), PHP_BINARY_READ)) {
    echo socket_strerror(socket_last_error());
}
var_dump($data);

/* Close sockets */
socket_close($sockets[0]);
socket_close($sockets[1]);
?>
```

Ejemplo 2. socket_create_pair() IPC example

```php
<?php
$ary = array();
$strone = 'Message From Parent.';
$strtwo = 'Message From Child.';
if (!socket_create_pair(AF_UNIX, SOCK_STREAM, 0, $ary)) {
    echo socket_strerror(socket_last_error());
}
$pid = pcntl_fork();
if ($pid == -1) {
    echo 'Could not fork Process.';
} elseif ($pid) {
    /*parent*/
    socket_close($ary[0]);
    if (!socket_write($ary[1], $strone, strlen($strone))) {
        echo socket_strerror(socket_last_error());
    }
    if (socket_read($ary[1], strlen($strtwo), PHP_BINARY_READ) == $strtwo) {
        echo "Recieved $strtwo\n";
    }
    socket_close($ary[1]);
} else {
    /*child*/
    socket_close($ary[1]);
    if (!socket_write($ary[0], $strtwo, strlen($strtwo))) {
        echo socket_strerror(socket_last_error());
    }
    if (socket_read($ary[0], strlen($strone), PHP_BINARY_READ) == $strone) {
        echo "Recieved $strone\n";
    }
    socket_close($ary[0]);
}
?>
```

socket_create

(PHP 4 >= 4.1.0, PHP 5)

socket_create -- Create a socket (endpoint for communication)

Description

resource **socket_create** (int domain, int type, int protocol)

Creates and returns a socket resource, also referred to as an endpoint of communication. A typical network connection is made up of 2 sockets, one performing the role of the client, and another performing the role of the server.

The *domain* parameter specifies the protocol family to be used by the socket.

Tabla 1. Available address/protocol families

Domain	Description
AF_INET	IPv4 Internet based protocols. TCP and UDP are common protocols of this protocol family.
AF_INET6	IPv6 Internet based protocols. TCP and UDP are common protocols of this protocol family. Support added in PHP 5.0.0.
AF_UNIX	Local communication protocol family. High efficiency and low overhead make it a great form of IPC (Interprocess Communication).

The *type* parameter selects the type of communication to be used by the socket.

Tabla 2. Available socket types

Type	Description
SOCK_STREAM	Provides sequenced, reliable, full-duplex, connection-based byte streams. An out-of-band data transmission mechanism may be supported. The TCP protocol is based on this socket type.
SOCK_DGRAM	Supports datagrams (connectionless, unreliable messages of a fixed maximum length). The UDP protocol is based on this socket type.
SOCK_SEQPACKET	Provides a sequenced, reliable, two-way connection-based data transmission path for datagrams of fixed maximum length; a consumer is required to read an entire packet with each read call.
SOCK_RAW	Provides raw network protocol access. This special type of socket can be used to manually construct any type of protocol. A common use for this socket type is to perform ICMP requests (like ping, traceroute, etc).
SOCK_RDM	Provides a reliable datagram layer that does not guarantee ordering. This is most likely not implemented on your operating system.

The *protocol* parameter sets the specific protocol within the specified *domain* to be used when communicating on the returned socket. The proper value can be retrieved by name by using **getprotobyname()**. If the desired protocol is TCP, or UDP the corresponding constants **SOL_TCP**, and **SOL_UDP** can also be used.

Tabla 3. Common protocols

Name	Description
icmp	The Internet Control Message Protocol is used primarily by gateways and hosts to report errors in datagram communication. The "ping" command (present in most modern operating systems) is an example application of ICMP.
udp	The User Datagram Protocol is a connectionless, unreliable, protocol with fixed record lengths. Due to these aspects, UDP requires a minimum amount of protocol overhead.
tcp	The Transmission Control Protocol is a reliable, connection based, stream oriented, full duplex protocol. TCP guarantees that all data packets will be received in the order in which they were sent. If any packet is somehow lost during communication, TCP will automatically retransmit the packet until the destination host acknowledges that packet. For reliability and performance reasons, the TCP implementation itself decides the appropriate octet boundaries of the underlying datagram communication layer. Therefore, TCP applications must allow for the possibility of partial record transmission.

socket_create() Returns a socket resource on success, or **FALSE** on error. The actual error code can be retrieved by calling **socket_last_error()**. This error code may be passed to **socket_strerror()** to get a textual explanation of the error.

> **Nota:** If an invalid *domain* or *type* is given, **socket_create()** defaults to **AF_INET** and **SOCK_STREAM** respectively and additionally emits an **E_WARNING** message.

See also **socket_accept()**, **socket_bind()**, **socket_connect()**, **socket_listen()**, **socket_last_error()**, and **socket_strerror()**.

socket_get_option

(PHP 4 >= 4.3.0, PHP 5)

socket_get_option -- Gets socket options for the socket

Description

mixed **socket_get_option** (resource socket, int level, int optname)

The **socket_get_option()** function retrieves the value for the option specified by the *optname* parameter for the socket specified by the *socket* parameter. **socket_get_option()** will return **FALSE** on failure.

The *level* parameter specifies the protocol level at which the option resides. For example, to retrieve options at the socket level, a *level* parameter of SOL_SOCKET would be used. Other levels, such as TCP, can be used by specifying the protocol number of that level. Protocol numbers can be found by using the **getprotobyname()** function.

Tabla 1. Available Socket Options

Option	Description
SO_DEBUG	Reports whether debugging information is being recorded.
SO_ACCEPTCONN	Reports whether socket listening is enabled.
SO_BROADCAST	Reports whether transmission of broadcast messages is supported.
SO_REUSEADDR	Reports whether local addresses can be reused.
SO_KEEPALIVE	Reports whether connections are kept active with periodic transmission of messages. If the connected socket fails to respond to these messages, the connection is broken and processes writing to that socket are notified with a SIGPIPE signal.
SO_LINGER	Reports whether the *socket* lingers on **socket_close()** if data is present.
SO_OOBINLINE	Reports whether the *socket* leaves out-of-band data inline.
SO_SNDBUF	Reports send buffer size information.
SO_RCVBUF	Reports recieve buffer size information.
SO_ERROR	Reports information about error status and clears it.
SO_TYPE	Reports the *socket* type.
SO_DONTROUTE	Reports whether outgoing messages bypass the standard routing facilities.
SO_RCVLOWAT	Reports the minimum number of bytes to process for *socket* input operations. (Defaults to 1)
SO_RCVTIMEO	Reports the timeout value for input operations.
SO_SNDLOWAT	Reports the minimum number of bytes to process for *socket* output operations.

Option	Description
SO_SNDTIMEO	Reports the timeout value specifying the amount of time that an output function blocks because flow control prevents data from being sent.

Nota: This function used to be called *socket_getopt()* prior to PHP 4.3.0

socket_getpeername

(PHP 4 >= 4.1.0, PHP 5)

socket_getpeername -- Queries the remote side of the given socket which may either result in host/port or in a Unix filesystem path, dependent on its type

Description

bool **socket_getpeername** (resource socket, string &addr [, int &port])

If the given socket is of type **AF_INET** or **AF_INET6**, **socket_getpeername()** will return the peers (remote) *IP address* in appropriate notation (e.g. *127.0.0.1* or *fe80::1*) in the *address* parameter and, if the optional *port* parameter is present, also the associated port.

If the given socket is of type **AF_UNIX**, **socket_getpeername()** will return the Unix filesystem path (e.g. */var/run/daemon.sock*) in the *address* parameter.

> **Nota: socket_getpeername()** should not be used with **AF_UNIX** sockets created with **socket_accept()**. Only sockets created with **socket_connect()** or a primary server socket following a call to **socket_bind()** will return meaningful values.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo. **socket_getpeername()** may also return **FALSE** if the socket type is not any of **AF_INET**, **AF_INET6**, or **AF_UNIX**, in which case the last socket error code is *not* updated.

See also **socket_getsockname()**, **socket_last_error()** and **socket_strerror()**.

socket_getsockname

(PHP 4 >= 4.1.0, PHP 5)

socket_getsockname -- Queries the local side of the given socket which may either result in host/port or in a Unix filesystem path, dependent on its type

Description

bool **socket_getsockname** (resource socket, string &addr [, int &port])

If the given socket is of type **AF_INET** or **AF_INET6**, **socket_getsockname()** will return the local *IP address* in appropriate notation (e.g. *127.0.0.1* or *fe80::1*) in the *address* parameter and, if the optional *port* parameter is present, also the associated port.

If the given socket is of type **AF_UNIX**, **socket_getsockname()** will return the Unix filesystem path (e.g. */var/run/daemon.sock*) in the *address* parameter.

> **Nota: socket_getsockname()** should not be used with **AF_UNIX** sockets created with socket_connect(). Only sockets created with socket_accept() or a primary server socket following a call to socket_bind() will return meaningful values.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo. **socket_getsockname()** may also return **FALSE** if the socket type is not any of **AF_INET**, **AF_INET6**, or **AF_UNIX**, in which case the last socket error code is *not* updated.

See also socket_getpeername(), socket_last_error() and socket_strerror().

socket_last_error

(PHP 4 >= 4.1.0, PHP 5)

socket_last_error -- Returns the last error on the socket

Description

int **socket_last_error** ([resource socket])

This function returns a socket error code.

If a socket resource is passed to this function, the last error which occurred on this particular socket is returned. If the socket resource is omitted, the error code of the last failed socket function is returned. The latter is in particular helpful for functions like socket_create() which don't return a socket on failure and socket_select() which can fail for reasons not directly tied to a particular socket. The error code is suitable to be fed to socket_strerror() which returns a string describing the given error code.

```
<?php
if (false == ($socket = @socket_create(AF_INET, SOCK_STREAM, SOL_TCP))) {
    die("Couldn't create socket, error code is: " . socket_last_error() .
        ",error message is: " . socket_strerror(socket_last_error()));
}
?>
```

> **Nota: socket_last_error()** does not clear the error code, use socket_clear_error() for this purpose.

socket_listen

(PHP 4 >= 4.1.0, PHP 5)

socket_listen -- Listens for a connection on a socket

Description

bool **socket_listen** (resource socket [, int backlog])

After the socket *socket* has been created using socket_create() and bound to a name with socket_bind(), it may be told to listen for incoming connections on *socket*.

A maximum of *backlog* incoming connections will be queued for processing. If a connection request arrives with the queue full the client may receive an error with an indication of

ECONNREFUSED, or, if the underlying protocol supports retransmission, the request may be ignored so that retries may succeed.

> **Nota:** The maximum number passed to the *backlog* parameter highly depends on the underlying platform. On Linux, it is silently truncated to **SOMAXCONN**. On win32, if passed **SOMAXCONN**, the underlying service provider responsible for the socket will set the backlog to a maximum *reasonable* value. There is no standard provision to find out the actual backlog value on this platform.

socket_listen() is applicable only to sockets of type **SOCK_STREAM** or **SOCK_SEQPACKET**.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo. The error code can be retrieved with **socket_last_error()**. This code may be passed to **socket_strerror()** to get a textual explanation of the error.

See also **socket_accept()**, **socket_bind()**, **socket_connect()**, **socket_create()** and **socket_strerror()**.

socket_read

(PHP 4 >= 4.1.0, PHP 5)

socket_read -- Reads a maximum of length bytes from a socket

Description

string **socket_read** (resource socket, int length [, int type])

The function **socket_read()** reads from the socket resource *socket* created by the **socket_create()** or **socket_accept()** functions. The maximum number of bytes read is specified by the *length* parameter. Otherwise you can use \r, \n, or \0 to end reading (depending on the *type* parameter, see below).

socket_read() returns the data as a string on success, or **FALSE** on error (including if the remote host has closed the connection). The error code can be retrieved with **socket_last_error()**. This code may be passed to **socket_strerror()** to get a textual representation of the error.

> **Nota: socket_read()** returns a zero length string ("") when there is no more data to read.

Optional *type* parameter is a named constant:

- PHP_BINARY_READ - use the system *recv()* function. Safe for reading binary data. (Default in PHP >= 4.1.0)
- PHP_NORMAL_READ - reading stops at \n or \r. (Default in PHP <= 4.0.6)

See also **socket_accept()**, **socket_bind()**, **socket_connect()**, **socket_listen()**, **socket_last_error()**, **socket_strerror()** and **socket_write()**.

socket_recv

(PHP 4 >= 4.1.0, PHP 5)

socket_recv -- Receives data from a connected socket

Description

int **socket_recv** (resource socket, string &buf, int len, int flags)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

socket_recvfrom

(PHP 4 >= 4.1.0, PHP 5)

socket_recvfrom -- Receives data from a socket, connected or not

Description

int **socket_recvfrom** (resource socket, string &buf, int len, int flags, string &name [, int &port])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

socket_recvfrom() has been binary safe since PHP 4.3.0

socket_select

(PHP 4 >= 4.1.0, PHP 5)

socket_select -- Runs the select() system call on the given arrays of sockets with a specified timeout

Description

int **socket_select** (array &read, array &write, array &except, int tv_sec [, int tv_usec])

socket_select() accepts arrays of sockets and waits for them to change status. Those coming with BSD sockets background will recognize that those socket resource arrays are in fact the so-called file descriptor sets. Three independent arrays of socket resources are watched.

The sockets listed in the *read* array will be watched to see if characters become available for reading (more precisely, to see if a read will not block - in particular, a socket resource is also ready on end-of-file, in which case a **socket_read()** will return a zero length string).

The sockets listed in the *write* array will be watched to see if a write will not block.

The sockets listed in the *except* array will be watched for exceptions.

Aviso
On exit, the arrays are modified to indicate which socket resource actually changed status.

You do not need to pass every array to **socket_select()**. You can leave it out and use an empty array or **NULL** instead. Also do not forget that those arrays are passed *by reference* and will be modified after **socket_select()** returns.

Ejemplo 1. socket_select() example

```
<?php
/* Prepare the read array */
$read = array($socket1, $socket2);

$num_changed_sockets = socket_select($read, $write = NULL, $except = NULL, 0);

if ($num_changed_sockets === false) {
    /* Error handling */
} else if ($num_changed_sockets > 0) {
    /* At least at one of the sockets something interesting happened */
}
?>
```

Nota: Due a limitation in the current Zend Engine it is not possible to pass a constant modifier like **NULL** directly as a parameter to a function which expects this parameter to be passed by reference. Instead use a temporary variable or an expression with the leftmost member being a temporary variable:

Ejemplo 2. Using NULL with socket_select()

```
<?php
socket_select($r, $w, $e = NULL, 0);
?>
```

The *tv_sec* and *tv_usec* together form the *timeout* parameter. The *timeout* is an upper bound on the amount of time elapsed before **socket_select()** return. *tv_sec* may be zero , causing **socket_select()** to return immediately. This is useful for polling. If *tv_sec* is **NULL** (no timeout), **socket_select()** can block indefinitely.

On success **socket_select()** returns the number of socket resources contained in the modified arrays, which may be zero if the timeout expires before anything interesting happens. On error **FALSE** is returned. The error code can be retrieved with socket_last_error().

Nota: Be sure to use the === operator when checking for an error. Since the **socket_select()** may return 0 the comparison with == would evaluate to **TRUE**:

Ejemplo 3. Understanding socket_select()'s result

```
<?php
if (false === socket_select($r, $w, $e = NULL, 0)) {
    echo "socket_select() failed, reason: " .
        socket_strerror(socket_last_error()) . "\n";
}
?>
```

Nota: Be aware that some socket implementations need to be handled very carefully. A few basic rules:

- You should always try to use **socket_select()** without timeout. Your program should have nothing to do if there is no data available. Code that depends on timeouts is not usually portable and difficult to debug.
- No socket resource must be added to any set if you do not intend to check its result after the **socket_select()** call, and respond appropriately. After

socket_select() returns, all socket resources in all arrays must be checked. Any socket resource that is available for writing must be written to, and any socket resource available for reading must be read from.

- If you read/write to a socket returns in the arrays be aware that they do not necessarily read/write the full amount of data you have requested. Be prepared to even only be able to read/write a single byte.
- It's common to most socket implementations that the only exception caught with the *except* array is out-of-bound data received on a socket.

See also **socket_read()**, **socket_write()**, **socket_last_error()** and **socket_strerror()**.

socket_send

(PHP 4 >= 4.1.0, PHP 5)

socket_send -- Sends data to a connected socket

Description

int **socket_send** (resource socket, string buf, int len, int flags)

The function **socket_send()** sends *len* bytes to the socket *socket* from *buf*

The value of *flags* can be any *ORed* combination of the following:

Tabla 1. possible values for *flags*

0x1	Process OOB (out-of-band) data
0x2	Peek at incoming message
0x4	Bypass routing, use direct interface
0x8	Data completes record
0x100	Data completes transaction

See also **socket_sendmsg()** and **socket_sendto()**.

socket_sendto

(PHP 4 >= 4.1.0, PHP 5)

socket_sendto -- Sends a message to a socket, whether it is connected or not

Description

int **socket_sendto** (resource socket, string buf, int len, int flags, string addr [, int port])

The function **socket_sendto()** sends *len* bytes from *buf* through the socket *socket* to the *port* at the address *addr*

The value of *flags* can be one of the following:

Tabla 1. possible values for *flags*

0x1	Process OOB (out-of-band) data.
0x2	Peek at incoming message.
0x4	Bypass routing, use direct interface.
0x8	Data completes record.
0x100	Data completes transaction.

Ejemplo 1. socket_sendto() Example

```
<?php
    $sh = socket_create(AF_INET, SOCK_STREAM, SOL_TCP);
    if (socket_bind($sh, '127.0.0.1', 4242)) {
        echo "Socket bound correctly";
    }
    $buf = 'Test Message';
    $len = strlen($buf);
    if (socket_sendto($sh, $buf, $len, 0x100, '192.168.0.2', 4242) !== false) {
        echo "Message sent correctly";
    }
    socket_close($sh);
?>
```

See also **socket_send()** and **socket_sendmsg()**.

socket_set_block

(PHP 4 >= 4.2.0, PHP 5)

socket_set_block -- Sets blocking mode on a socket resource

Description

bool **socket_set_block** (resource socket)

The **socket_set_block()** function removes the O_NONBLOCK flag on the socket specified by the *socket* parameter.

Ejemplo 1. socket_set_block() example

```
<?php

$port = 9090;
if (!$socket = socket_create_listen($port)) {
    echo socket_strerror(socket_last_error());
}

if (!socket_set_option($socket, SOL_SOCKET, SO_REUSEADDR, 1)) {
    echo socket_strerror(socket_last_error());
}

if (!socket_set_nonblock($socket)) { // $socket is now nonblocking
    echo socket_strerror(socket_last_error());
}

if (!socket_set_block($socket)) {     // $socket is now blocking
    echo socket_strerror(socket_last_error());
}

?>
```

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

See also **socket_set_nonblock()** and **socket_set_option()**

socket_set_nonblock

(PHP 4 >= 4.1.0, PHP 5)

socket_set_nonblock -- Sets nonblocking mode for file descriptor fd

Description

bool **socket_set_nonblock** (resource socket)

The **socket_set_nonblock()** function sets the O_NONBLOCK flag on the socket specified by the *socket* parameter.

Ejemplo 1. socket_set_nonblock() example

```
<?php
$port = 9090;
if (!$socket = socket_create_listen($port)) {
    echo socket_strerror(socket_last_error());
}

if (!socket_set_option($socket, SOL_SOCKET, SO_REUSEADDR, 1)) {
    echo socket_strerror(socket_last_error());
}

if (!socket_set_nonblock($socket)) {
    echo socket_strerror(socket_last_error());
}
?>
```

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

See also **socket_set_block()** and **socket_set_option()**

socket_set_option

(PHP 4 >= 4.3.0, PHP 5)

socket_set_option -- Sets socket options for the socket

Description

bool **socket_set_option** (resource socket, int level, int optname, mixed optval)

The **socket_set_option()** function sets the option specified by the *optname* parameter, at the protocol level specified by the *level* parameter, to the value pointed to by the *optval* parameter for the socket specified by the *socket* parameter. **socket_set_option()** will return **FALSE** on failure.

The *level* parameter specifies the protocol level at which the option resides. For example, to retrieve options at the socket level, a *level* parameter of SOL_SOCKET would be used. Other levels, such as TCP, can be used by specifying the protocol number of that level. Protocol numbers can be found by using the **getprotobyname()** function.

The available socket options are the same as those for the **socket_get_option()** function.

> **Nota:** This function used to be called *socket_setopt()* prior to PHP 4.3.0

socket_shutdown

(PHP 4 >= 4.1.0, PHP 5)

socket_shutdown -- Shuts down a socket for receiving, sending, or both

Description

bool **socket_shutdown** (resource socket [, int how])

The **socket_shutdown()** function allows you to stop incoming, outgoing or all data (the default) from being sent through the *socket*

The value of *how* can be one of the following:

Tabla 1. possible values for *how*

0	Shutdown socket reading
1	Shutdown socket writing
2	Shutdown socket reading and writing

socket_strerror

(PHP 4 >= 4.1.0, PHP 5)

socket_strerror -- Return a string describing a socket error

Description

string **socket_strerror** (int errno)

socket_strerror() takes as its *errno* parameter a socket error code as returned by **socket_last_error()** and returns the corresponding explanatory text. This makes it a bit more pleasant to figure out why something didn't work; for instance, instead of having to track down a system include file to find out what '-111' means, you just pass it to **socket_strerror()**, and it tells you what happened.

Ejemplo 1. socket_strerror() example

```
<?php
if (false == ($socket = @socket_create(AF_INET, SOCK_STREAM, SOL_TCP))) {
   echo "socket_create() failed: reason: " . socket_strerror(socket_last_error()) . "\n
}

if (false == (@socket_bind($socket, '127.0.0.1', 80))) {
   echo "socket_bind() failed: reason: " . socket_strerror(socket_last_error($socket))
}
?>
```

The expected output from the above example (assuming the script is not run with root privileges):

```
socket_bind() failed: reason: Permission denied
```

See also **socket_accept()**, **socket_bind()**, **socket_connect()**, **socket_listen()**, and **socket_create()**.

socket_write

(PHP 4 >= 4.1.0, PHP 5)

socket_write -- Write to a socket

Description

int **socket_write** (resource socket, string buffer [, int length])

The function **socket_write()** writes to the socket *socket* from *buffer*.

The optional parameter *length* can specify an alternate length of bytes written to the socket. If this length is greater then the buffer length, it is silently truncated to the length of the buffer.

Returns the number of bytes successfully written to the socket or **FALSE** one error. The error code can be retrieved with **socket_last_error()**. This code may be passed to **socket_strerror()** to get a textual explanation of the error.

> **Nota: socket_write()** does not necessarily write all bytes from the given buffer. It's valid that, depending on the network buffers etc., only a certain amount of data, even one byte, is written though your buffer is greater. You have to watch out so you don't unintentionally forget to transmit the rest of your data.
>
> **Nota:** It is perfectly valid for **socket_write()** to return zero which means no bytes have been written. Be sure to use the === operator to check for **FALSE** in case of an error.

See also **socket_accept()**, **socket_bind()**, **socket_connect()**, **socket_listen()**, **socket_read()** and

socket_strerror().

CXVII. Standard PHP Library (SPL) Functions

Introducción

SPL is a collection of interfaces and classes that are meant to solve standard problems.

> **Sugerencia:** A more detailed documentation of SPL can be found here.

Instalación

This extension is available and compiled by default in PHP 5.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

`RIT_LEAVES_ONLY` (**integer**)

`RIT_SELF_FIRST` (**integer**)

`RIT_CHILD_FIRST` (**integer**)

`CIT_CALL_TOSTRING` (**integer**)

`CIT_CATCH_GET_CHILD` (**integer**)

Tabla de contenidos

CachingIterator::next -- Move the iterator forward
CachingIterator::rewind -- Rewind the iterator
CachingIterator::__toString -- Return the string representation of the current element
CachingIterator::valid -- Check whether the current element is valid
CachingRecursiveIterator::getChildren -- Return the inner iterator's children as a CachingRecursiveIterator
CachingRecursiveIterator::hasChildren -- Check whether the current element of the inner iterator has children
DirectoryIterator::__construct -- Constructs a new dir iterator from a path
DirectoryIterator::current -- Return this (needed for Iterator interface)
DirectoryIterator::getATime -- Get last access time of file
DirectoryIterator::getCTime -- Get inode modification time of file
DirectoryIterator::getChildren -- Returns an iterator for the current entry if it is a directory
DirectoryIterator::getFilename -- Return filename of current dir entry
DirectoryIterator::getGroup -- Get file group
DirectoryIterator::getInode -- Get file inode
DirectoryIterator::getMTime -- Get last modification time of file
DirectoryIterator::getOwner -- Get file owner
DirectoryIterator::getPath -- Return directory path
DirectoryIterator::getPathname -- Return path and filename of current dir entry
DirectoryIterator::getPerms -- Get file permissions
DirectoryIterator::getSize -- Get file size
DirectoryIterator::getType -- Get file type
DirectoryIterator::isDir -- Returns true if file is directory
DirectoryIterator::isDot -- Returns true if current entry is '.' or '..'
DirectoryIterator::isExecutable -- Returns true if file is executable
DirectoryIterator::isFile -- Returns true if file is a regular file
DirectoryIterator::isLink -- Returns true if file is symbolic link
DirectoryIterator::isReadable -- Returns true if file can be read
DirectoryIterator::isWritable -- Returns true if file can be written
DirectoryIterator::key -- Return current dir entry
DirectoryIterator::next -- Move to next entry
DirectoryIterator::rewind -- Rewind dir back to the start
DirectoryIterator::valid -- Check whether dir contains more entries
FilterIterator::current -- Get the current element value
FilterIterator::getInnerIterator -- Get the inner iterator
FilterIterator::key -- Get the current key
FilterIterator::next -- Move the iterator forward
FilterIterator::rewind -- Rewind the iterator
FilterIterator::valid -- Check whether the current element is valid
LimitIterator::getPosition -- Return the current position
LimitIterator::next -- Move the iterator forward
LimitIterator::rewind -- Rewind the iterator to the specified starting offset
LimitIterator::seek -- Seek to the given position
LimitIterator::valid -- Check whether the current element is valid
ParentIterator::getChildren -- Return the inner iterator's children contained in a ParentIterator
ParentIterator::hasChildren -- Check whether the inner iterator's current element has children
ParentIterator::next -- Move the iterator forward
ParentIterator::rewind -- Rewind the iterator
RecursiveDirectoryIterator::getChildren -- Returns an iterator for the current entry if it is a directory
RecursiveDirectoryIterator::hasChildren -- Returns whether current entry is a directory and not '.' or '..'
RecursiveDirectoryIterator::key -- Return path and filename of current dir entry
RecursiveDirectoryIterator::next -- Move to next entry
RecursiveDirectoryIterator::rewind -- Rewind dir back to the start

RecursiveIteratorIterator::current -- Access the current element value
RecursiveIteratorIterator::getDepth -- Get the current depth of the recursive iteration
RecursiveIteratorIterator::getSubIterator -- The current active sub iterator
RecursiveIteratorIterator::key -- Access the current key
RecursiveIteratorIterator::next -- Move forward to the next element
RecursiveIteratorIterator::rewind -- Rewind the iterator to the first element of the top level inner iterator
RecursiveIteratorIterator::valid -- Check whether the current position is valid
SimpleXMLIterator::current -- Return current SimpleXML entry
SimpleXMLIterator::getChildren -- Returns an iterator for the current entry if it is a SimpleXML object
SimpleXMLIterator::hasChildren -- Returns whether current entry is a SimpleXML object
SimpleXMLIterator::key -- Return current SimpleXML key
SimpleXMLIterator::next -- Move to next entry
SimpleXMLIterator::rewind -- Rewind SimpleXML back to the start
SimpleXMLIterator::valid -- Check whether SimpleXML contains more entries
class_implements -- Return the interfaces which are implemented by the given class
class_parents -- Return the parent classes of the given class
iterator_count -- Count the elements in an iterator
iterator_to_array -- Copy the iterator into an array
spl_classes -- Return available SPL classes

ArrayIterator::current

(no version information, might be only in CVS)

ArrayIterator::current -- Return current array entry

Description

mixed **ArrayIterator::current** (void)

This function returns the current array entry

Ejemplo 1. ArrayIterator::current() example

```
<?php
$array = array('1' => 'one',
               '2' => 'two',
               '3' => 'three');

$arrayobject = new ArrayObject($array);
$iterator = $arrayobject->getIterator();

for($iterator = $arrayobject->getIterator();
    $iterator->valid();
    $iterator->next()) {

    echo $iterator->key() . ' => ' . $iterator->current() . "\n";
}
?>
```

The above example will output:

```
1 => one
2 => two
3 => three
```

ArrayIterator::key

(no version information, might be only in CVS)

ArrayIterator::key -- Return current array key

Description

mixed **ArrayIterator::key** (void)

This function returns the current array key

Ejemplo 1. ArrayIterator::key() example

```
<?php
$array = array('key' => 'value');

$arrayobject = new ArrayObject($array);
$iterator = $arrayobject->getIterator();

echo $iterator->key(); //key
?>
```

ArrayIterator::next

(no version information, might be only in CVS)

ArrayIterator::next -- Move to next entry

Description

void **ArrayIterator::next** (void)

This function moves the iterator to the next entry.

Ejemplo 1. ArrayIterator::next() example

```
<?php
$arrayobject = new ArrayObject();

$arrayobject[] = 'zero';
$arrayobject[] = 'one';

$iterator = $arrayobject->getIterator();

while($iterator->valid()) {
    echo $iterator->key() . ' => ' . $iterator->current() . "\n";

    $iterator->next();
}
?>
```

The above example will output:

```
0 => zero
1 => one
```

ArrayIterator::rewind

(no version information, might be only in CVS)

ArrayIterator::rewind -- Rewind array back to the start

Description

void **ArrayIterator::rewind** (void)

This function rewinds the iterator to the beginning.

Ejemplo 1. ArrayIterator::rewind() example

```
<?php
$arrayobject = new ArrayObject();

$arrayobject[] = 'zero';
$arrayobject[] = 'one';
$arrayobject[] = 'two';

$iterator = $arrayobject->getIterator();

$iterator->next();
echo $iterator->key(); //1

$iterator->rewind(); //rewinding to the begining
echo $iterator->key(); //0
?>
```

ArrayIterator::seek

(no version information, might be only in CVS)

ArrayIterator::seek -- Seek to position

Description

void **ArrayIterator::seek** (int position)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ArrayIterator::valid

(no version information, might be only in CVS)

ArrayIterator::valid -- Check whether array contains more entries

Description

bool **ArrayIterator::valid** (void)

This function checks if the array contains any more entries.

Ejemplo 1. ArrayIterator::valid() example

```
<?php
$array = array('1' => 'one');

$arrayobject = new ArrayObject($array);
$iterator = $arrayobject->getIterator();

var_dump($iterator->valid()); //bool(true)

$iterator->next(); // advance to the next item

//bool(false) because there is only one array element
var_dump($iterator->valid());
?>
```

ArrayObject::append

(no version information, might be only in CVS)

ArrayObject::append -- Appends the value

Description

void **ArrayObject::append** (mixed newval)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ArrayObject::__construct

(no version information, might be only in CVS)

ArrayObject::__construct -- Construct a new array object

Description

ArrayObject **ArrayObject::__construct** (mixed input)

This constructs a new array object. The *input* parameter accepts an array or another ArrayObject.

Ejemplo 1. ArrayObject::__construct() example

```
<?php
$array = array('1' => 'one',
               '2' => 'two',
               '3' => 'three');

$arrayobject = new ArrayObject($array);

var_dump($arrayobject);
?>
```

The above example will output:

```
object(ArrayObject)#1 (3) {
  [1]=>
  string(3) "one"
  [2]=>
  string(3) "two"
  [3]=>
  string(5) "three"
}
```

ArrayObject::count

(no version information, might be only in CVS)

ArrayObject::count -- Return the number of elements in the Iterator

Description

int **ArrayObject::count** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ArrayObject::getIterator

(no version information, might be only in CVS)

ArrayObject::getIterator -- Create a new iterator from an ArrayObject instance

Description

ArrayIterator **ArrayObject::getIterator** (void)

This function will return an iterator from an ArrayObject.

Ejemplo 1. ArrayObject::getIterator() example

```
<?php
$array = array('1' => 'one',
               '2' => 'two',
               '3' => 'three');

$arrayobject = new ArrayObject($array);

$iterator = $arrayobject->getIterator();

while($iterator->valid()) {
    echo $iterator->key() . ' => ' . $iterator->current() . "\n";

    $iterator->next();
}
?>
```

The above example will output:

```
1 => one
2 => two
3 => three
```

ArrayObject::offsetExists

(no version information, might be only in CVS)

ArrayObject::offsetExists -- Returns whether the requested $index exists

Description

bool **ArrayObject::offsetExists** (mixed index)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ArrayObject::offsetGet

(no version information, might be only in CVS)

ArrayObject::offsetGet -- Returns the value at the specified $index

Description

bool **ArrayObject::offsetGet** (mixed index)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ArrayObject::offsetSet

(no version information, might be only in CVS)

ArrayObject::offsetSet -- Sets the value at the specified $index to $newval

Description

void **ArrayObject::offsetSet** (mixed index, mixed newval)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ArrayObject::offsetUnset

(no version information, might be only in CVS)

ArrayObject::offsetUnset -- Unsets the value at the specified $index

Description

void **ArrayObject::offsetUnset** (mixed index)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

CachingIterator::hasNext

(no version information, might be only in CVS)

CachingIterator::hasNext -- Check whether the inner iterator has a valid next element

Description

boolean **CachingIterator::hasNext** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

CachingIterator::next

(no version information, might be only in CVS)

CachingIterator::next -- Move the iterator forward

Description

void **CachingIterator::next** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

CachingIterator::rewind

(no version information, might be only in CVS)

CachingIterator::rewind -- Rewind the iterator

Description

void **CachingIterator::rewind** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

CachingIterator::__toString

(no version information, might be only in CVS)

CachingIterator::__toString -- Return the string representation of the current element

Description

string **CachingIterator::__toString** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

CachingIterator::valid

(no version information, might be only in CVS)

CachingIterator::valid -- Check whether the current element is valid

Description

boolean **CachingIterator::valid** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

CachingRecursiveIterator::getChildren

(no version information, might be only in CVS)

CachingRecursiveIterator::getChildren -- Return the inner iterator's children as a CachingRecursiveIterator

Description

CachingRecursiveIterator **CachingRecursiveIterator::getChildren** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

CachingRecursiveIterator::hasChildren

(no version information, might be only in CVS)

CachingRecursiveIterator::hasChildren -- Check whether the current element of the inner iterator has children

Description

bolean **CachingRecursiveIterator::hasChildren** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

DirectoryIterator::__construct

(no version information, might be only in CVS)

DirectoryIterator::__construct -- Constructs a new dir iterator from a path

Description

DirectoryIterator **DirectoryIterator::__construct** (string path)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

DirectoryIterator::current

(no version information, might be only in CVS)

DirectoryIterator::current -- Return this (needed for Iterator interface)

Description

DirectoryIterator **DirectoryIterator::current** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

DirectoryIterator::getATime

(no version information, might be only in CVS)

DirectoryIterator::getATime -- Get last access time of file

Description

int **DirectoryIterator::getATime** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

DirectoryIterator::getCTime

(no version information, might be only in CVS)

DirectoryIterator::getCTime -- Get inode modification time of file

Description

int **DirectoryIterator::getCTime** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

DirectoryIterator::getChildren

(no version information, might be only in CVS)

DirectoryIterator::getChildren -- Returns an iterator for the current entry if it is a directory

Description

RecursiveDirectoryIterator **DirectoryIterator::getChildren** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

DirectoryIterator::getFilename

(no version information, might be only in CVS)

DirectoryIterator::getFilename -- Return filename of current dir entry

Description

string **DirectoryIterator::getFilename** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

DirectoryIterator::getGroup

(no version information, might be only in CVS)

DirectoryIterator::getGroup -- Get file group

Description

int **DirectoryIterator::getGroup** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

DirectoryIterator::getInode

(no version information, might be only in CVS)

DirectoryIterator::getInode -- Get file inode

Description

int **DirectoryIterator::getInode** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

DirectoryIterator::getMTime

(no version information, might be only in CVS)

DirectoryIterator::getMTime -- Get last modification time of file

Description

int **DirectoryIterator::getMTime** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

DirectoryIterator::getOwner

(no version information, might be only in CVS)

DirectoryIterator::getOwner -- Get file owner

Description

int **DirectoryIterator::getOwner** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

DirectoryIterator::getPath

(no version information, might be only in CVS)

DirectoryIterator::getPath -- Return directory path

Description

string **DirectoryIterator::getPath** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

DirectoryIterator::getPathname

(no version information, might be only in CVS)

DirectoryIterator::getPathname -- Return path and filename of current dir entry

Description

string **DirectoryIterator::getPathname** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

DirectoryIterator::getPerms

(no version information, might be only in CVS)

DirectoryIterator::getPerms -- Get file permissions

Description

int **DirectoryIterator::getPerms** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

DirectoryIterator::getSize

(no version information, might be only in CVS)

DirectoryIterator::getSize -- Get file size

Description

int **DirectoryIterator::getSize** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

DirectoryIterator::getType

(no version information, might be only in CVS)

DirectoryIterator::getType -- Get file type

Description

string **DirectoryIterator::getType** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

DirectoryIterator::isDir

(no version information, might be only in CVS)

DirectoryIterator::isDir -- Returns true if file is directory

Description

bool **DirectoryIterator::isDir** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

DirectoryIterator::isDot

(no version information, might be only in CVS)

DirectoryIterator::isDot -- Returns true if current entry is '.' or '..'

Description

bool **DirectoryIterator::isDot** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

DirectoryIterator::isExecutable

(no version information, might be only in CVS)

DirectoryIterator::isExecutable -- Returns true if file is executable

Description

bool **DirectoryIterator::isExecutable** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

DirectoryIterator::isFile

(no version information, might be only in CVS)

DirectoryIterator::isFile -- Returns true if file is a regular file

Description

bool **DirectoryIterator::isFile** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

DirectoryIterator::isLink

(no version information, might be only in CVS)

DirectoryIterator::isLink -- Returns true if file is symbolic link

Description

bool **DirectoryIterator::isLink** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

DirectoryIterator::isReadable

(no version information, might be only in CVS)

DirectoryIterator::isReadable -- Returns true if file can be read

Description

bool **DirectoryIterator::isReadable** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

DirectoryIterator::isWritable

(no version information, might be only in CVS)

DirectoryIterator::isWritable -- Returns true if file can be written

Description

bool **DirectoryIterator::isWritable** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

DirectoryIterator::key

(no version information, might be only in CVS)

DirectoryIterator::key -- Return current dir entry

Description

string **DirectoryIterator::key** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

DirectoryIterator::next

(no version information, might be only in CVS)

DirectoryIterator::next -- Move to next entry

Description

void **DirectoryIterator::next** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

DirectoryIterator::rewind

(no version information, might be only in CVS)

DirectoryIterator::rewind -- Rewind dir back to the start

Description

void **DirectoryIterator::rewind** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

DirectoryIterator::valid

(no version information, might be only in CVS)

DirectoryIterator::valid -- Check whether dir contains more entries

Description

string **DirectoryIterator::valid** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

FilterIterator::current

(no version information, might be only in CVS)

FilterIterator::current -- Get the current element value

Description

mixed **FilterIterator::current** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

FilterIterator::getInnerIterator

(no version information, might be only in CVS)

FilterIterator::getInnerIterator -- Get the inner iterator

Description

Iterator **FilterIterator::getInnerIterator** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

FilterIterator::key

(no version information, might be only in CVS)

FilterIterator::key -- Get the current key

Description

mixed **FilterIterator::key** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

FilterIterator::next

(no version information, might be only in CVS)

FilterIterator::next -- Move the iterator forward

Description

void **FilterIterator::next** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

FilterIterator::rewind

(no version information, might be only in CVS)

FilterIterator::rewind -- Rewind the iterator

Description

void **FilterIterator::rewind** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

FilterIterator::valid

(no version information, might be only in CVS)

FilterIterator::valid -- Check whether the current element is valid

Description

boolean **FilterIterator::valid** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

LimitIterator::getPosition

(no version information, might be only in CVS)

LimitIterator::getPosition -- Return the current position

Description

int **LimitIterator::getPosition** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

LimitIterator::next

(no version information, might be only in CVS)

LimitIterator::next -- Move the iterator forward

Description

void **LimitIterator::next** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

LimitIterator::rewind

(no version information, might be only in CVS)

LimitIterator::rewind -- Rewind the iterator to the specified starting offset

Description

void **LimitIterator::rewind** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

LimitIterator::seek

(no version information, might be only in CVS)

LimitIterator::seek -- Seek to the given position

Description

void **LimitIterator::seek** (int position)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

LimitIterator::valid

(no version information, might be only in CVS)

LimitIterator::valid -- Check whether the current element is valid

Description

boolean **LimitIterator::valid** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ParentIterator::getChildren

(no version information, might be only in CVS)

ParentIterator::getChildren -- Return the inner iterator's children contained in a ParentIterator

Description

ParentIterator **ParentIterator::getChildren** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ParentIterator::hasChildren

(no version information, might be only in CVS)

ParentIterator::hasChildren -- Check whether the inner iterator's current element has children

Description

boolean **ParentIterator::hasChildren** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ParentIterator::next

(no version information, might be only in CVS)

ParentIterator::next -- Move the iterator forward

Description

void **ParentIterator::next** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

ParentIterator::rewind

(no version information, might be only in CVS)

ParentIterator::rewind -- Rewind the iterator

Description

void **ParentIterator::rewind** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

RecursiveDirectoryIterator::getChildren

(no version information, might be only in CVS)

RecursiveDirectoryIterator::getChildren -- Returns an iterator for the current entry if it is a directory

Description

object **RecursiveDirectoryIterator::getChildren** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

RecursiveDirectoryIterator::hasChildren

(no version information, might be only in CVS)

RecursiveDirectoryIterator::hasChildren -- Returns whether current entry is a directory and not '.' or '..'

Description

bool **RecursiveDirectoryIterator::hasChildren** ([bool allow_links])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

RecursiveDirectoryIterator::key

(no version information, might be only in CVS)

RecursiveDirectoryIterator::key -- Return path and filename of current dir entry

Description

string **RecursiveDirectoryIterator::key** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

RecursiveDirectoryIterator::next

(no version information, might be only in CVS)

RecursiveDirectoryIterator::next -- Move to next entry

Description

void **RecursiveDirectoryIterator::next** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

RecursiveDirectoryIterator::rewind

(no version information, might be only in CVS)

RecursiveDirectoryIterator::rewind -- Rewind dir back to the start

Description

void **RecursiveDirectoryIterator::rewind** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

RecursiveIteratorIterator::current

(no version information, might be only in CVS)

RecursiveIteratorIterator::current -- Access the current element value

Description

mixed **RecursiveIteratorIterator::current** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

RecursiveIteratorIterator::getDepth

(no version information, might be only in CVS)

RecursiveIteratorIterator::getDepth -- Get the current depth of the recursive iteration

Description

int **RecursiveIteratorIterator::getDepth** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

RecursiveIteratorIterator::getSubIterator

(no version information, might be only in CVS)

RecursiveIteratorIterator::getSubIterator -- The current active sub iterator

Description

RecursiveIterator **RecursiveIteratorIterator::getSubIterator** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

RecursiveIteratorIterator::key

(no version information, might be only in CVS)

RecursiveIteratorIterator::key -- Access the current key

Description

mixed **RecursiveIteratorIterator::key** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

RecursiveIteratorIterator::next

(no version information, might be only in CVS)

RecursiveIteratorIterator::next -- Move forward to the next element

Description

void **RecursiveIteratorIterator::next** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

RecursiveIteratorIterator::rewind

(no version information, might be only in CVS)

RecursiveIteratorIterator::rewind -- Rewind the iterator to the first element of the top level inner iterator

Description

void **RecursiveIteratorIterator::rewind** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

RecursiveIteratorIterator::valid

(no version information, might be only in CVS)

RecursiveIteratorIterator::valid -- Check whether the current position is valid

Description

bolean **RecursiveIteratorIterator::valid** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

SimpleXMLIterator::current

(no version information, might be only in CVS)

SimpleXMLIterator::current -- Return current SimpleXML entry

Description

mixed **SimpleXMLIterator::current** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

SimpleXMLIterator::getChildren

(no version information, might be only in CVS)

SimpleXMLIterator::getChildren -- Returns an iterator for the current entry if it is a SimpleXML object

Description

object **SimpleXMLIterator::getChildren** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

SimpleXMLIterator::hasChildren

(no version information, might be only in CVS)

SimpleXMLIterator::hasChildren -- Returns whether current entry is a SimpleXML object

Description

bool **SimpleXMLIterator::hasChildren** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

SimpleXMLIterator::key

(no version information, might be only in CVS)

SimpleXMLIterator::key -- Return current SimpleXML key

Description

mixed **SimpleXMLIterator::key** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

SimpleXMLIterator::next

(no version information, might be only in CVS)

SimpleXMLIterator::next -- Move to next entry

Description

void **SimpleXMLIterator::next** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

SimpleXMLIterator::rewind

(no version information, might be only in CVS)

SimpleXMLIterator::rewind -- Rewind SimpleXML back to the start

Description

void **SimpleXMLIterator::rewind** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

SimpleXMLIterator::valid

(no version information, might be only in CVS)

SimpleXMLIterator::valid -- Check whether SimpleXML contains more entries

Description

bool **SimpleXMLIterator::valid** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

class_implements

(PHP 5)

class_implements -- Return the interfaces which are implemented by the given class

Description

array **class_implements** (object class)

This function returns an array with the name of the interfaces that the given *class* implements.

Ejemplo 1. class_implements() example

```
<?php

interface foo { }
class bar implements foo {}

print_r(class_implements(new bar));

?>
```

El resultado del ejemplo seria:

```
Array
(
    [foo] => foo
)
```

class_parents

(PHP 5)

class_parents -- Return the parent classes of the given class

Description

array **class_parents** (object class)

This function returns an array with the name of the parent classes of the given *class*.

Ejemplo 1. class_parents() example

```
<?php

class foo { }
class bar extends foo {}

print_r(class_parents(new bar));

?>
```

El resultado del ejemplo seria:

```
Array
(
    [foo] => foo
)
```

iterator_count

(no version information, might be only in CVS)

iterator_count -- Count the elements in an iterator

Description

int **iterator_count** (IteratorAggregate iterator)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

iterator_to_array

(no version information, might be only in CVS)

iterator_to_array -- Copy the iterator into an array

Description

array **iterator_to_array** (IteratorAggregate iterator)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

spl_classes

(PHP 5)

spl_classes -- Return available SPL classes

Description

array **spl_classes** (void)

This function returns an array with the current available SPL classes.

Ejemplo 1. spl_classes() example

```
<?php

print_r(spl_classes());

?>
```

El resultado del ejemplo seria algo similar a:

```
Array
(
    [ArrayObject] => ArrayObject
    [ArrayIterator] => ArrayIterator
    [CachingIterator] => CachingIterator
    [CachingRecursiveIterator] => CachingRecursiveIterator
    [DirectoryIterator] => DirectoryIterator
    [FilterIterator] => FilterIterator
    [LimitIterator] => LimitIterator
    [ParentIterator] => ParentIterator
    [RecursiveDirectoryIterator] => RecursiveDirectoryIterator
    [RecursiveIterator] => RecursiveIterator
    [RecursiveIteratorIterator] => RecursiveIteratorIterator
    [SeekableIterator] => SeekableIterator
    [SimpleXMLIterator] => SimpleXMLIterator
)
```

CXVIII. SQLite Functions

Introducción

This is an extension for the SQLite Embeddable SQL Database Engine. SQLite is a C library that implements an embeddable SQL database engine. Programs that link with the SQLite library can have SQL database access without running a separate RDBMS process.

SQLite is not a client library used to connect to a big database server. SQLite is the server. The SQLite library reads and writes directly to and from the database files on disk.

> **Nota:** For further information see the SQLite Website (http://sqlite.org/).

Installation

Read the INSTALL file, which comes with the package. Or just use the PEAR installer with "pear install sqlite". SQLite itself is already included, You do not need to install any additional software.

Windows users may download the DLL version of the SQLite extension here: (php_sqlite.dll).

In PHP 5, the SQLite extension and the engine itself are bundled and compiled by default.

> **Windows installation for unprivileged accounts:** On Windows operating systems, unprivileged accounts don't have the *TMP* environment variable set by default. This will make sqlite create temporary files in the windows directory, which is not desirable. So, you should set the *TMP* environment variable for the web server or the user account the web server is running under. If Apache is your web server, you can accomplish this via a **SetEnv** directive in your `httpd.conf` file. For example:
>
> ```
> SetEnv TMP c:/temp
> ```
>
> If you are unable to establish this setting at the server level, you can implement the setting in your script:
>
> ```
> putenv('TMP=C:/temp');
> ```
>
> The setting must refer to a directory that the web server has permission to create files in and subsequently write to and delete the files it created. Otherwise, you may receive the following error message: `malformed database schema - unable to open a temporary database file for storing temporary tables`

Requirimientos

In order to have these functions available, you must compile PHP with SQLite support, or load the SQLite extension dynamically from your `php.ini`.

Tipos de recursos

There are two resources used in the SQLite Interface. The first one is the database connection, the second one the result set.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

The functions **sqlite_fetch_array()** and **sqlite_current()** use a constant for the different types of result arrays. The following constants are defined:

SQLite result type constants

`SQLITE_ASSOC` (**int**)

> Columns are returned into the array having the field name as the array index.

`SQLITE_BOTH` (**int**)

Columns are returned into the array having both a numerical index and the field name as the array index.

SQLITE_NUM (int)

Columns are returned into the array having a numerical index to the fields. This index starts with 0, the first field in the result.

A number of functions may return status codes. The following constants are defined:

SQLite status code constants

SQLITE_OK (int)

Successful result.

SQLITE_ERROR (int)

SQL error or missing database.

SQLITE_INTERNAL (int)

An internal logic error in SQLite.

SQLITE_PERM (int)

Access permission denied.

SQLITE_ABORT (int)

Callback routine requested an abort.

SQLITE_BUSY (int)

The database file is locked.

SQLITE_LOCKED (int)

A table in the database is locked.

SQLITE_NOMEM (int)

Memory allocation failed.

SQLITE_READONLY (int)

Attempt to write a readonly database.

SQLITE_INTERRUPT (int)

Operation terminated internally.

SQLITE_IOERR (int)

Disk I/O error occurred.

SQLITE_CORRUPT (int)

The database disk image is malformed.

SQLITE_NOTFOUND (int)

(Internal) Table or record not found.

SQLITE_FULL (int)

Insertion failed because database is full.

SQLITE_CANTOPEN (int)

Unable to open the database file.

SQLITE_PROTOCOL (int)

Database lock protocol error.

SQLITE_EMPTY (int)

(Internal) Database table is empty.

SQLITE_SCHEMA (int)

The database schema changed.

SQLITE_TOOBIG (int)

Too much data for one row of a table.

SQLITE_CONSTRAINT (int)

Abort due to constraint violation.

SQLITE_MISMATCH (int)

Data type mismatch.

SQLITE_MISUSE (int)

Library used incorrectly.

SQLITE_NOLFS (int)

Uses of OS features not supported on host.

SQLITE_AUTH (int)

Authorized failed.

`SQLITE_ROW` (**int**)

Internal process has another row ready.

`SQLITE_DONE` (**int**)

Internal process has finished executing.

Clases predefinidas

SQLiteDatabase

Represents an opened SQLite database.

Constructor

- __construct - construct a new SQLiteDatabase object

Métodos

- query - Execute a query
- queryExec - Execute a result-less query
- arrayQuery - Execute a query and return the result as an array
- singleQuery - Execute a query and return either an array for one single column or the value of the first row
- unbufferedQuery - Execute an unbuffered query
- lastInsertRowid - Returns the rowid of the most recently inserted row
- changes - Returns the number of rows changed by the most recent statement
- createAggregate - Register an aggregating UDF for use in SQL statements
- createFunction - Register a UDF for use in SQL statements
- busyTimeout - Sets or disables busy timeout duration
- lastErorr - Returns the last error code of the most recently encountered error
- fetchColumnTypes - Return an array of column types from a particular table

---

SQLiteResult

Represents a buffered SQLite result set.

Métodos

- fetch - Fetches the next row from the result set as an array
- fetchObject - Fetches the next row from the result set as an object
- fetchSingle - Fetches the first column from the result set as a string
- fetchAll - Fetches all rows from the result set as an array of arrays
- column - Fetches a column from the current row of the result set
- numFields - Returns the number of fields in the result set
- fieldName - Returns the name of a particular field in the result set
- current - Fetches the current row from the result set as an array
- key - Return the current row index
- next - Seek to the next row number
- valid - Returns whether more rows are available
- rewind - Seek to the first row number of the result set
- prev - Seek to the previous row number of the result set
- hasPrev - Returns whether or not a previous row is available
- numRows - Returns the number of rows in the result set
- seek - Seek to a particular row number

SQLiteUnbuffered

Represents an unbuffered SQLite result set. Unbuffered results sets are sequential, forward-seeking only.

Métodos

- fetch - Fetches the next row from the result set as an array
- fetchObject - Fetches the next row from the result set as an object

- fetchSingle - Fetches the first column from the result set as a string
- fetchAll - Fetches all rows from the result set as an array of arrays
- column - Fetches a column from the current row of the result set
- numFields - Returns the number of fields in the result set
- fieldName - Returns the name of a particular field in the result set
- current - Fetches the current row from the result set as an array
- next - Seek to the next row number
- valid - Returns whether more rows are available

Configuración en tiempo de ejecución

El comportamiento de estas funciones está afectado por los valores definidos en `php.ini`.

Tabla 1. SQLite Configure Options

Name	Default	Changeable
sqlite.assoc_case	0	PHP_INI_ALL

For further details and definitions of the PHP_INI_* constants, see the Apéndice H.

A continuación se presenta una corta explicación de las directivas de configuración.

sqlite.assoc_case **int**

Whether to use mixed case (*0*), upper case (*1*) or lower case (*2*) hash indexes.

This option is primarily useful when you need compatibility with other database systems, where the names of the columns are always returned as uppercase or lowercase, regardless of the case of the actual field names in the database schema.

The SQLite library returns the column names in their natural case (that matches the case you used in your schema). When *sqlite.assoc_case* is set to *0* the natural case will be preserved. When it is set to *1* or *2*, PHP will apply case folding on the hash keys to upper- or lower-case the keys, respectively.

Use of this option incurs a slight performance penalty, but is MUCH faster than performing the case folding yourself using PHP script.

Tabla de contenidos

sqlite_create_aggregate -- Register an aggregating UDF for use in SQL statements
sqlite_create_function -- Registers a "regular" User Defined Function for use in SQL statements
sqlite_current -- Fetches the current row from a result set as an array
sqlite_error_string -- Returns the textual description of an error code
sqlite_escape_string -- Escapes a string for use as a query parameter
sqlite_exec -- Executes a result-less query against a given database
sqlite_factory -- Opens a SQLite database and returns a SQLiteDatabase object
sqlite_fetch_all -- Fetches all rows from a result set as an array of arrays
sqlite_fetch_array -- Fetches the next row from a result set as an array
sqlite_fetch_column_types -- Return an array of column types from a particular table
sqlite_fetch_object -- Fetches the next row from a result set as an object
sqlite_fetch_single -- Fetches the first column of a result set as a string
sqlite_fetch_string -- Alias of **sqlite_fetch_single()**
sqlite_field_name -- Returns the name of a particular field
sqlite_has_more -- Finds whether or not more rows are available
sqlite_has_prev -- Returns whether or not a previous row is available
sqlite_key -- Returns the current row index
sqlite_last_error -- Returns the error code of the last error for a database
sqlite_last_insert_rowid -- Returns the rowid of the most recently inserted row
sqlite_libencoding -- Returns the encoding of the linked SQLite library
sqlite_libversion -- Returns the version of the linked SQLite library
sqlite_next -- Seek to the next row number
sqlite_num_fields -- Returns the number of fields in a result set
sqlite_num_rows -- Returns the number of rows in a buffered result set
sqlite_open -- Opens a SQLite database and create the database if it does not exist
sqlite_popen -- Opens a persistent handle to an SQLite database and create the database if it does not exist
sqlite_prev -- Seek to the previous row number of a result set
sqlite_query -- Executes a query against a given database and returns a result handle
sqlite_rewind -- Seek to the first row number
sqlite_seek -- Seek to a particular row number of a buffered result set
sqlite_single_query -- Executes a query and returns either an array for one single column or the value of the first row
sqlite_udf_decode_binary -- Decode binary data passed as parameters to an UDF
sqlite_udf_encode_binary -- Encode binary data before returning it from an UDF
sqlite_unbuffered_query -- Execute a query that does not prefetch and buffer all data
sqlite_valid -- Returns whether more rows are available

sqlite_array_query

(PHP 5)

sqlite_array_query

(no version information, might be only in CVS)

SQLiteDatabase->arrayQuery -- Execute a query against a given database and returns an array

Descripción

array **sqlite_array_query** (resource dbhandle, string query [, int result_type [, bool decode_binary]])

array **sqlite_array_query** (string query, resource dbhandle [, int result_type [, bool decode_binary]])

Object oriented style (method):

class **SQLiteDatabase** {

array **arrayQuery** (string query [, int result_type [, bool decode_binary]])

}

sqlite_array_query() executes the given query and returns an array of the entire result set. It is similar to calling **sqlite_query()** and then **sqlite_fetch_array()** for each row in the result set. **sqlite_array_query()** is significantly faster than the aforementioned.

> **Sugerencia: sqlite_array_query()** is best suited to queries returning 45 rows or less. If you have more data than that, it is recommended that you write your scripts to use **sqlite_unbuffered_query()** instead for more optimal performance.

Lista de parámetros

query

> The query to be executed.

dbhandle

> The SQLite Database resource; returned from **sqlite_open ()** when used procedurally. This parameter is not required when using the object-oriented method.

result_type

> The optional *result_type* parameter accepts a constant and determines how the returned array will be indexed. Using **SQLITE_ASSOC** will return only associative indices (named fields) while **SQLITE_NUM** will return only numerical indices (ordinal field numbers). **SQLITE_BOTH** will return both associative and numerical indices. **SQLITE_BOTH** is the default for this function.

decode_binary

> When the *decode_binary* parameter is set to **TRUE** (the default), PHP will decode the binary encoding it applied to the data if it was encoded using the **sqlite_escape_string()**. You should normally leave this value at its default, unless you are interoperating with databases created by other sqlite capable applications.
>
> **Nota:** Two alternative syntaxes are supported for compatibility with other database extensions (such as MySQL). The preferred form is the first, where the *dbhandle* parameter is the first parameter to the function.

Valores retornados

Returns an array of the entire result set; **FALSE** otherwise.

The column names returned by **SQLITE_ASSOC** and **SQLITE_BOTH** will be case-folded according to the value of the sqlite.assoc_case configuration option.

Ejemplos

Ejemplo 1. Procedural style

```
<?php
$dbhandle = sqlite_open('sqlitedb');
$result = sqlite_array_query($dbhandle, 'SELECT name, email FROM users LIMIT 25', SQLIT
foreach ($result as $entry) {
    echo 'Name: ' . $entry['name'] . '  E-mail: ' . $entry['email'];
}
?>
```

Ejemplo 2. Object-oriented style

```
<?php
$dbhandle = new SQLiteDatabase('sqlitedb');
$result = $dbhandle->arrayQuery('SELECT name, email FROM users LIMIT 25', SQLITE_ASSOC)
foreach ($result as $entry) {
    echo 'Name: ' . $entry['name'] . '  E-mail: ' . $entry['email'];
}
?>
```

Ver también

sqlite_query()
sqlite_fetch_array()
sqlite_fetch_string()

sqlite_busy_timeout

(PHP 5)

sqlite_busy_timeout

(no version information, might be only in CVS)

SQLiteDatabase->busyTimeout -- Set busy timeout duration, or disable busy handlers

Descripción

void **sqlite_busy_timeout** (resource dbhandle, int milliseconds)

Object oriented style (method):

class **SQLiteDatabase** {

void **busyTimeout** (int milliseconds)

}

Set the maximum time, in milliseconds, that SQLite will wait for a *dbhandle* to become ready for use.

Lista de parámetros

dbhandle

The SQLite Database resource; returned from **sqlite_open ()** when used procedurally. This parameter is not required when using the object-oriented method.

milliseconds

The number of milliseconds. When set to *0*, busy handlers will be disabled and SQLite will return immediately with a *SQLITE_BUSY* status code if another process/thread has the database locked for an update.

PHP sets the default busy timeout to be 60 seconds when the database is opened.

Nota: There are one thousand (1000) milliseconds in one second.

Ejemplos

Ejemplo 1. Procedural style

```
<?php
$dbhandle = sqlite_open('sqlitedb');
sqlite_busy_timeout($dbhandle, 10000); // set timeout to 10 seconds
sqlite_busy_timeout($dbhandle, 0); // disable busy handler
?>
```

Ejemplo 2. Object oriented style

```
<?php
$dbhandle = new SQLiteDatabase('sqlitedb');
$dbhandle->busyTimeout(10000); // 10 seconds
$dbhandle->busyTimeout(0); // disable
?>
```

Ver también

sqlite_open()

sqlite_changes

(PHP 5)

sqlite_changes

(no version information, might be only in CVS)

SQLiteDatabase->changes -- Returns the number of rows that were changed by the most recent SQL statement

Descripción

int **sqlite_changes** (resource dbhandle)

Object oriented style (method):

class **SQLiteDatabase** {

int **changes** (void)

}

Returns the numbers of rows that were changed by the most recent SQL statement executed against the *dbhandle* database handle.

Lista de parámetros

dbhandle

The SQLite Database resource; returned from **sqlite_open ()** when used procedurally. This parameter is not required when using the object-oriented method.

Ejemplos

Ejemplo 1. Procedural style

```
<?php
$dbhandle = sqlite_open('mysqlitedb');
$query = sqlite_query($dbhandle, "UPDATE users SET email='jDoe@example.com' WHERE usern
if (!$query) {
    exit('Error in query.');
} else {
    echo 'Number of rows modified: ', sqlite_changes($dbhandle);
}
?>
```

Ejemplo 2. Object oriented style

```
<?php
$dbhandle = new SQLiteDatabase('mysqlitedb');
$query = $dbhandle->query("UPDATE users SET email='jDoe@example.com' WHERE username='jD
if (!$query) {
    exit('Error in query.');
} else {
    echo 'Number of rows modified: ', $dbhandle->changes();
}
?>
```

Ver también

sqlite_open()

sqlite_close

(PHP 5)

sqlite_close -- Closes an open SQLite database

Descripción

void **sqlite_close** (resource dbhandle)

Closes the given *database* handle. If the database was persistent, it will be closed and removed from

the persistent list.

Lista de parámetros

dbhandle

The SQLite Database resource; returned from **sqlite_open ()** when used procedurally.

Ejemplos

Ejemplo 1. sqlite_close() example

```
<?php
$dbhandle = sqlite_open('sqlitedb');
sqlite_close($dbhandle);
?>
```

Ver también

sqlite_open()
sqlite_popen()

sqlite_column

(PHP 5)

sqlite_column

(no version information, might be only in CVS)

SQLiteResult->column

(no version information, might be only in CVS)

SQLiteUnbuffered->column -- Fetches a column from the current row of a result set

Descripción

mixed **sqlite_column** (resource result, mixed index_or_name [, bool decode_binary])

class **SQLiteResult** {

mixed **column** (mixed index_or_name [, bool decode_binary])

}class **SQLiteUnbuffered** {

mixed **column** (mixed index_or_name [, bool decode_binary])

}

Fetches the value of a column named *index_or_name* (if it is a string), or of the ordinal column numbered *index_or_name* (if it is an integer) from the current row of the query result handle *result*.

Lista de parámetros

result

The SQLite result resource. This parameter is not required when using the object-oriented method.

index_or_name

The column index or name to fetch.

decode_binary

When the *decode_binary* parameter is set to **TRUE** (the default), PHP will decode the binary encoding it applied to the data if it was encoded using the **sqlite_escape_string()**. You should normally leave this value at its default, unless you are interoperating with databases created by other sqlite capable applications.

Notes

Nota: Use this function when you are iterating a large result set with many columns, or with columns that contain large amounts of data.

Ver también

sqlite_fetch_string()

sqlite_create_aggregate

(PHP 5)

sqlite_create_aggregate

(no version information, might be only in CVS)

SQLiteDatabase->createAggregate -- Register an aggregating UDF for use in SQL statements

Descripción

bool **sqlite_create_aggregate** (resource dbhandle, string function_name, callback step_func, callback finalize_func [, int num_args])

Object oriented style (method):

class **SQLiteDatabase** {

bool **createAggregate** (string function_name, callback step_func, callback finalize_func [, int num_args])

}

sqlite_create_aggregate() is similar to **sqlite_create_function()** except that it registers functions that can be used to calculate a result aggregated across all the rows of a query.

The key difference between this function and **sqlite_create_function()** is that two functions are required to manage the aggregate; *step_func* is called for each row of the result set. Your PHP function should accumulate the result and store it into the aggregation context. Once all the rows have been processed, *finalize_func* will be called and it should then take the data from the aggregation context and return the result. Callback functions should return a type understood by SQLite (i.e. scalar type).

Lista de parámetros

dbhandle

The SQLite Database resource; returned from **sqlite_open ()** when used procedurally. This parameter is not required when using the object-oriented method.

function_name

The name of the function used in SQL statements.

step_func

Callback function called for each row of the result set.

finalize_func

Callback function to aggregate the "stepped" data from each row.

num_args

Hint to the SQLite parser if the callback function accepts a predetermined number of arguments.

Ejemplos

Ejemplo 1. max_length aggregation function example

```
<?php
$data = array(
   'one',
   'two',
   'three',
   'four',
   'five',
   'six',
   'seven',
   'eight',
   'nine',
   'ten',
   );
$dbhandle = sqlite_open(':memory:');
sqlite_query($dbhandle, "CREATE TABLE strings(a)");
foreach ($data as $str) {
    $str = sqlite_escape_string($str);
    sqlite_query($dbhandle, "INSERT INTO strings VALUES ('$str')");
}

function max_len_step(&$context, $string)
{
    if (strlen($string) > $context) {
        $context = strlen($string);
    }
}

function max_len_finalize(&$context)
{
    return $context;
}

sqlite_create_aggregate($dbhandle, 'max_len', 'max_len_step', 'max_len_finalize');

var_dump(sqlite_array_query($dbhandle, 'SELECT max_len(a) from strings'));

?>
```

In this example, we are creating an aggregating function that will calculate the length of the longest string in one of the columns of the table. For each row, the *max_len_step* function is called and passed a *context* parameter. The context parameter is just like any other PHP variable and be set to hold an array or even an object value. In this example, we are simply using it to hold the maximum length we have seen so far; if the *string* has a length longer than the current maximum, we update the context to hold this new maximum length.

After all of the rows have been processed, SQLite calls the *max_len_finalize* function to determine the aggregate result. Here, we could perform some kind of calculation based on the data found in the *context*. In our simple example though, we have been calculating the result as the query progressed, so we simply need to return the context value.

> **Nota:** The example above will not work correctly if the column contains binary data. Take a look at the manual page for **sqlite_udf_decode_binary()** for an explanation of why this is so, and an example of how to make it respect the binary encoding.
>
> **Sugerencia:** It is NOT recommended for you to store a copy of the values in the context and then process them at the end, as you would cause SQLite to use a lot of memory to process the query - just think of how much memory you would need if a million rows were stored in memory, each containing a string 32 bytes in length.
>
> **Sugerencia:** You can use **sqlite_create_function()** and **sqlite_create_aggregate()** to override SQLite native SQL functions.

Ver también

sqlite_create_function()
sqlite_udf_encode_binary()
sqlite_udf_decode_binary()

sqlite_create_function

(PHP 5)

sqlite_create_function

(no version information, might be only in CVS)

SQLiteDatabase->createFunction -- Registers a "regular" User Defined Function for use in SQL statements

Descripción

bool **sqlite_create_function** (resource dbhandle, string function_name, callback callback [, int num_args])

Object oriented style (method):

class **SQLiteDatabase** {

bool **createFunction** (string function_name, callback callback [, int num_args])

}

sqlite_create_function() allows you to register a PHP function with SQLite as an UDF (User Defined Function), so that it can be called from within your SQL statements.

The UDF can be used in any SQL statement that can call functions, such as SELECT and UPDATE statements and also in triggers.

Lista de parámetros

dbhandle

The SQLite Database resource; returned from **sqlite_open ()** when used procedurally. This parameter is not required when using the object-oriented method.

function_name

The name of the function used in SQL statements.

callback

Callback function to handle the defined SQL function.

Nota: Callback functions should return a type understood by SQLite (i.e. scalar type).

num_args

Hint to the SQLite parser if the callback function accepts a predetermined number of arguments.

Nota: Two alternative syntaxes are supported for compatibility with other database extensions (such as MySQL). The preferred form is the first, where the *dbhandle* parameter is the first parameter to the function.

Ejemplos

Ejemplo 1. sqlite_create_function() example

```
<?php
function md5_and_reverse($string)
{
    return strrev(md5($string));
}

if ($dbhandle = sqlite_open('mysqlitedb', 0666, $sqliteerror)) {

    sqlite_create_function($dbhandle, 'md5rev', 'md5_and_reverse', 1);

    $sql  = 'SELECT md5rev(filename) FROM files';
    $rows = sqlite_array_query($dbhandle, $sql);
} else {
    echo 'Error opening sqlite db: ' . $sqliteerror;
    exit;
}
?>
```

In this example, we have a function that calculates the md5 sum of a string, and then reverses it. When the SQL statement executes, it returns the value of the filename transformed by our function. The data returned in *$rows* contains the processed result.

The beauty of this technique is that you do not need to process the result using a foreach() loop after you have queried for the data.

PHP registers a special function named *php* when the database is first opened. The php function can be used to call any PHP function without having to register it first.

Ejemplo 2. Example of using the PHP function

```
<?php
$rows = sqlite_array_query($dbhandle, "SELECT php('md5', filename) from files");
?>
```

This example will call the **md5()** on each *filename* column in the database and return the result into *$rows*

Nota: For performance reasons, PHP will not automatically encode/decode binary data passed to and from your UDF's. You need to manually encode/decode the parameters and return values if you need to process binary data in this way. Take a look at **sqlite_udf_encode_binary()** and **sqlite_udf_decode_binary()** for more details.

Sugerencia: It is not recommended to use UDF's to handle processing of binary data, unless high performance is not a key requirement of your application.

Sugerencia: You can use **sqlite_create_function()** and **sqlite_create_aggregate()** to

override SQLite native SQL functions.

Ver también

sqlite_create_aggregate()

sqlite_current

(PHP 5)

sqlite_current

(no version information, might be only in CVS)

SQLiteResult->current

(no version information, might be only in CVS)

SQLiteUnbuffered->current -- Fetches the current row from a result set as an array

Descripción

array **sqlite_current** (resource result [, int result_type [, bool decode_binary]])

Object oriented style (method):

class **SQLiteResult** {

array **current** ([int result_type [, bool decode_binary]])

}class **SQLiteUnbuffered** {

array **current** ([int result_type [, bool decode_binary]])

}

sqlite_current() is identical to **sqlite_fetch_array()** except that it does not advance to the next row prior to returning the data; it returns the data from the current position only.

Lista de parámetros

result

The SQLite result resource. This parameter is not required when using the object-oriented method.

result_type

The optional *result_type* parameter accepts a constant and determines how the returned array will be indexed. Using **SQLITE_ASSOC** will return only associative indices (named fields) while **SQLITE_NUM** will return only numerical indices (ordinal field numbers).

SQLITE_BOTH will return both associative and numerical indices. **SQLITE_BOTH** is the default for this function.

decode_binary

When the *decode_binary* parameter is set to **TRUE** (the default), PHP will decode the binary encoding it applied to the data if it was encoded using the **sqlite_escape_string()**. You should normally leave this value at its default, unless you are interoperating with databases created by other sqlite capable applications.

Valores retornados

Returns an array of the current row from a result set; **FALSE** if the current position is beyond the final row.

The column names returned by **SQLITE_ASSOC** and **SQLITE_BOTH** will be case-folded according to the value of the sqlite.assoc_case configuration option.

Ver también

sqlite_seek()
sqlite_next()
sqlite_fetch_array()

sqlite_error_string

(PHP 5)

sqlite_error_string -- Returns the textual description of an error code

Descripción

string **sqlite_error_string** (int error_code)

Returns a human readable description of the *error_code* returned from **sqlite_last_error()**.

Ver también

sqlite_last_error()

sqlite_escape_string

(PHP 5)

sqlite_escape_string -- Escapes a string for use as a query parameter

Descripción

string **sqlite_escape_string** (string item)

sqlite_escape_string() will correctly quote the string specified by *item* for use in an SQLite SQL statement. This includes doubling up single-quote characters (') and checking for binary-unsafe characters in the query string.

If the *item* contains a *NUL* character, or if it begins with a character whose ordinal value is *0x01*, PHP will apply a binary encoding scheme so that you can safely store and retrieve binary data.

Although the encoding makes it safe to insert the data, it will render simple text comparisons and *LIKE* clauses in your queries unusable for the columns that contain the binary data. In practice, this shouldn't be a problem, as your schema should be such that you don't use such things on binary columns (in fact, it might be better to store binary data using other means, such as in files).

Aviso
addslashes() should *NOT* be used to quote your strings for SQLite queries; it will lead to strange results when retrieving your data.

> **Nota:** Do not use this function to encode the return values from UDF's created using **sqlite_create_function()** or **sqlite_create_aggregate()** - use **sqlite_udf_encode_binary()** instead.

Ver también

sqlite_udf_encode_binary()

sqlite_exec

(no version information, might be only in CVS)

sqlite_exec

(no version information, might be only in CVS)

SQLiteDatabase->exec -- Executes a result-less query against a given database

Descripción

bool **sqlite_exec** (resource dbhandle, string query)

bool **sqlite_exec** (string query, resource dbhandle)

Object oriented style (method):

class **SQLiteDatabase** {

bool **exec** (string query)

}

Executes an SQL statement given by the *query* against a given database handle (specified by the *dbhandle* parameter).

Aviso
SQLite *will* execute multiple queries separated by semicolons, so you can use it to execute a batch of SQL that you have loaded from a file or have embedded in a script.

Lista de parámetros

query

The query to be executed.

dbhandle

The SQLite Database resource; returned from **sqlite_open ()** when used procedurally. This parameter is not required when using the object-oriented method.

Nota: Two alternative syntaxes are supported for compatibility with other database extensions (such as MySQL). The preferred form is the first, where the *dbhandle* parameter is the first parameter to the function.

Valores retornados

This function will return a boolean result; **`TRUE`** for success or **`FALSE`** for failure. If you need to run a query that returns rows, see **sqlite_query()**.

The column names returned by **`SQLITE_ASSOC`** and **`SQLITE_BOTH`** will be case-folded according to the value of the sqlite.assoc_case configuration option.

Ejemplos

Ejemplo 1. Procedural example

```
<?php
$dbhandle = sqlite_open('mysqlitedb');
$query = sqlite_exec($dbhandle, "UPDATE users SET email='jDoe@example.com' WHERE userna
if (!$query) {
    exit('Error in query.');
} else {
    echo 'Number of rows modified: ', sqlite_changes($dbhandle);
}
?>
```

Ejemplo 2. Object-oriented example

```
<?php
$dbhandle = new SQLiteDatabase('mysqlitedb');
$query = $dbhandle->exec("UPDATE users SET email='jDoe@example.com' WHERE username='jDo
if (!$query) {
    exit('Error in query.');
} else {
    echo 'Number of rows modified: ', $dbhandle->changes();
}
?>
```

Ver también

sqlite_query()
sqlite_unbuffered_query()

sqlite_array_query()

sqlite_factory

(PHP 5)

sqlite_factory -- Opens a SQLite database and returns a SQLiteDatabase object

Descripción

SQLiteDatabase **sqlite_factory** (string filename [, int mode [, string &error_message]])

sqlite_factory() behaves similarly to **sqlite_open()** in that it opens an SQLite database or attempts to create it if it does not exist. However, a SQLiteDatabase object is returned rather than a resource. Please see the **sqlite_open()** reference page for further usage and caveats.

Lista de parámetros

filename

The filename of the SQLite database.

mode

The mode of the file. Intended to be used to open the database in read-only mode. Presently, this parameter is ignored by the sqlite library. The default value for mode is the octal value *0666* and this is the recommended value.

error_message

Passed by reference and is set to hold a descriptive error message explaining why the database could not be opened if there was an error.

Valores retornados

Returns a SQLiteDatabase object on success, **NULL** on error.

Ejemplos

Ejemplo 1. sqlite_factory() example

```
<?php
$dbhandle = sqlite_factory('sqlitedb');
$dbhandle->query('SELECT user_id, username FROM users');

/* functionally ñalent to: */

$dbhandle = new SQLiteDatabase('sqlitedb');
$dbhandle->query('SELECT user_id, username FROM users');

?>
```

Ver también

sqlite_open()
sqlite_popen()

sqlite_fetch_all

(PHP 5)

sqlite_fetch_all

(no version information, might be only in CVS)

SQLiteResult->fetchAll

(no version information, might be only in CVS)

SQLiteUnbuffered->fetchAll -- Fetches all rows from a result set as an array of arrays

Descripción

array **sqlite_fetch_all** (resource result [, int result_type [, bool decode_binary]])

Object oriented style (method):

class **SQLiteResult** {

array **fetchAll** ([int result_type [, bool decode_binary]])

}class **SQLiteUnbuffered** {

array **fetchAll** ([int result_type [, bool decode_binary]])

}

sqlite_fetch_all() returns an array of the entire result set from the *result* resource. It is similar to calling **sqlite_query()** (or **sqlite_unbuffered_query()**) and then **sqlite_fetch_array()** for each row in the result set.

Lista de parámetros

result

The SQLite result resource. This parameter is not required when using the object-oriented method.

result_type

The optional *result_type* parameter accepts a constant and determines how the returned array will be indexed. Using **SQLITE_ASSOC** will return only associative indices (named fields) while **SQLITE_NUM** will return only numerical indices (ordinal field numbers).

SQLITE_BOTH will return both associative and numerical indices. **SQLITE_BOTH** is the default for this function.

decode_binary

When the *decode_binary* parameter is set to **TRUE** (the default), PHP will decode the binary encoding it applied to the data if it was encoded using the **sqlite_escape_string()**. You should normally leave this value at its default, unless you are interoperating with databases created by other sqlite capable applications.

Valores retornados

Returns an array of the current row from a result set; **FALSE** if the current position is beyond the final row.

The column names returned by **SQLITE_ASSOC** and **SQLITE_BOTH** will be case-folded according to the value of the sqlite.assoc_case configuration option.

Ejemplos

Ejemplo 1. Procedural example

```
<?php
$dbhandle = sqlite_open('sqlitedb');
$query = sqlite_query($dbhandle, 'SELECT name, email FROM users LIMIT 25');
$result = sqlite_fetch_all($query, SQLITE_ASSOC);
foreach ($result as $entry) {
    echo 'Name: ' . $entry['name'] . '  E-mail: ' . $entry['email'];
}
?>
```

Ejemplo 2. Object-oriented example

```
<?php
$dbhandle = new SQLiteDatabase('sqlitedb');

$query = $dbhandle->query('SELECT name, email FROM users LIMIT 25'); // buffered result
$query = $dbhandle->unbufferedQuery('SELECT name, email FROM users LIMIT 25'); // unbuf

$result = $query->fetchAll(SQLITE_ASSOC);
foreach ($result as $entry) {
    echo 'Name: ' . $entry['name'] . '  E-mail: ' . $entry['email'];
}
?>
```

Ver también

sqlite_fetch_array()

sqlite_fetch_array

(PHP 5)

sqlite_fetch_array

(no version information, might be only in CVS)

SQLiteResult->fetch

(no version information, might be only in CVS)

SQLiteUnbuffered->fetch -- Fetches the next row from a result set as an array

Descripción

array **sqlite_fetch_array** (resource result [, int result_type [, bool decode_binary]])

Object oriented style (method):

class **SQLiteResult** {

array **fetch** ([int result_type [, bool decode_binary]])

}class **SQLiteUnbuffered** {

array **fetch** ([int result_type [, bool decode_binary]])

}

Fetches the next row from the given *result* handle. If there are no more rows, returns **FALSE**, otherwise returns an associative array representing the row data.

Lista de parámetros

result

The SQLite result resource. This parameter is not required when using the object-oriented method.

result_type

The optional *result_type* parameter accepts a constant and determines how the returned array will be indexed. Using **SQLITE_ASSOC** will return only associative indices (named fields) while **SQLITE_NUM** will return only numerical indices (ordinal field numbers). **SQLITE_BOTH** will return both associative and numerical indices. **SQLITE_BOTH** is the default for this function.

decode_binary

When the *decode_binary* parameter is set to **TRUE** (the default), PHP will decode the binary encoding it applied to the data if it was encoded using the **sqlite_escape_string()**. You should normally leave this value at its default, unless you are interoperating with databases created by other sqlite capable applications.

Valores retornados

Returns an array of the next row from a result set; **FALSE** if the next position is beyond the final row.

The column names returned by **SQLITE_ASSOC** and **SQLITE_BOTH** will be case-folded according to the value of the sqlite.assoc_case configuration option.

Ejemplos

Ejemplo 1. Procedural example

```
<?php
$dbhandle = sqlite_open('sqlitedb');
$query = sqlite_query($dbhandle, 'SELECT name, email FROM users LIMIT 25');
$result = sqlite_fetch_all($query, SQLITE_ASSOC);
foreach ($result as $entry) {
    echo 'Name: ' . $entry['name'] . '  E-mail: ' . $entry['email'];
}
?>
```

Ejemplo 2. Object-oriented example

```
<?php
$dbhandle = new SQLiteDatabase('sqlitedb');

$query = $dbhandle->query('SELECT name, email FROM users LIMIT 25'); // buffered result
$query = $dbhandle->unbufferedQuery('SELECT name, email FROM users LIMIT 25'); // unbuf

$result = $query->fetchAll(SQLITE_ASSOC);
foreach ($result as $entry) {
    echo 'Name: ' . $entry['name'] . '  E-mail: ' . $entry['email'];
}
?>
```

Ver también

sqlite_array_query()
sqlite_fetch_string()

sqlite_fetch_column_types

(PHP 5)

sqlite_fetch_column_types

(no version information, might be only in CVS)

SQLiteDatabase->fetchColumnTypes -- Return an array of column types from a particular table

Descripción

array **sqlite_fetch_column_types** (string table_name, resource dbhandle [, int result_type])

Object oriented style (method):

class **SQLiteDatabase** {

array **fetchColumnTypes** (string table_name [, int result_type])

}

sqlite_fetch_column_types() returns an array of column data types from the specified *table_name* table.

Lista de parámetros

table_name

The table name to query.

dbhandle

The SQLite Database resource; returned from **sqlite_open ()** when used procedurally. This parameter is not required when using the object-oriented method.

result_type

The optional *result_type* parameter accepts a constant and determines how the returned array will be indexed. Using **SQLITE_ASSOC** will return only associative indices (named fields) while **SQLITE_NUM** will return only numerical indices (ordinal field numbers). **SQLITE_BOTH** will return both associative and numerical indices. **SQLITE_BOTH** is the default for this function.

Valores retornados

Returns an array of column data types; **FALSE** on error.

The column names returned by **SQLITE_ASSOC** and **SQLITE_BOTH** will be case-folded according to the value of the sqlite.assoc_case configuration option.

Ejemplos

Ejemplo 1. Procedural example

```
<?php
$db = sqlite_open('mysqlitedb');
sqlite_query($db, 'CREATE TABLE foo (bar varchar(10), arf text)');
$cols = sqlite_fetch_column_types('foo', $db, SQLITE_ASSOC);

foreach ($cols as $column => $type) {
    echo "Column: $column  Type: $type";
}
?>
```

Ejemplo 2. Object-oriented example

```
<?php
$db = new SQLiteDatabase('mysqlitedb');
$db->query('CREATE TABLE foo (bar varchar(10), arf text)');
$cols = $db->fetchColumnTypes('foo', SQLITE_ASSOC);

foreach ($cols as $column => $type) {
    echo "Column: $column  Type: $type";
}
?>
```

El resultado del ejemplo seria:

```
Column: bar  Type: VARCHAR
Column: arf  Type: TEXT
```

sqlite_fetch_object

(PHP 5)

sqlite_fetch_object

(no version information, might be only in CVS)

SQLiteResult->fetchObject

(no version information, might be only in CVS)

SQLiteUnbuffered->fetchObject -- Fetches the next row from a result set as an object

Descripción

object **sqlite_fetch_object** (resource result [, string class_name [, array ctor_params [, bool decode_binary]]])

Object oriented style (method):

class **SQLiteResult** {

object **fetchObject** ([string class_name [, array ctor_params [, bool decode_binary]]])

}class **SQLiteUnbuffered** {

object **fetchObject** ([string class_name [, array ctor_params [, bool decode_binary]]])

}

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

sqlite_fetch_single

(PHP 5)

sqlite_fetch_single

(no version information, might be only in CVS)

SQLiteResult->fetchSingle

(no version information, might be only in CVS)

SQLiteUnbuffered->fetchSingle -- Fetches the first column of a result set as a string

Descripción

string **sqlite_fetch_single** (resource result [, bool decode_binary])

Object oriented style (method):

class **SQLiteResult** {

string **fetchSingle** ([bool decode_binary])

}class **SQLiteUnbuffered** {

string **fetchSingle** ([bool decode_binary])

}

sqlite_fetch_single() is identical to **sqlite_fetch_array()** except that it returns the value of the first column of the rowset.

This is the most optimal way to retrieve data when you are only interested in the values from a single column of data.

Lista de parámetros

result

The SQLite result resource. This parameter is not required when using the object-oriented method.

decode_binary

When the *decode_binary* parameter is set to **TRUE** (the default), PHP will decode the binary encoding it applied to the data if it was encoded using the **sqlite_escape_string()**. You should normally leave this value at its default, unless you are interoperating with databases created by other sqlite capable applications.

Ejemplos

Ejemplo 1. A sqlite_fetch_single() example

```
<?php
if ($dbhandle = sqlite_open('mysqlitedb', 0666, $sqliteerror)) {

    $sql = "SELECT id FROM sometable WHERE id = 42";
    $res = sqlite_query($dbhandle, $sql);

    if (sqlite_num_rows($res) > 0) {
        echo sqlite_fetch_single($res); // 42
    }

    sqlite_close($dbhandle);
}
?>
```

Ver también

sqlite_fetch_array()

sqlite_fetch_string

sqlite_fetch_string -- Alias of **sqlite_fetch_single()**

Descripción

This function is an alias of **sqlite_fetch_single()**.

sqlite_field_name

(PHP 5)

sqlite_field_name

(no version information, might be only in CVS)

SQLiteResult->fieldName

(no version information, might be only in CVS)

SQLiteUnbuffered->fieldName -- Returns the name of a particular field

Descripción

string **sqlite_field_name** (resource result, int field_index)

Object oriented style (method):

class **SQLiteResult** {

string **fieldName** (int field_index)

}class **SQLiteUnbuffered** {

string **fieldName** (int field_index)

}

Given the ordinal column number, *field_index*, **sqlite_field_name()** returns the name of that field in the result set *result*.

Lista de parámetros

result

The SQLite result resource. This parameter is not required when using the object-oriented

method.

field_index

The ordinal column number in the result set.

Valores retornados

Returns the name of a field in an SQLite result set, given the ordinal column number; **FALSE** on error.

The column names returned by **SQLITE_ASSOC** and **SQLITE_BOTH** will be case-folded according to the value of the sqlite.assoc_case configuration option.

sqlite_has_more

(PHP 5)

sqlite_has_more -- Finds whether or not more rows are available

Descripción

bool **sqlite_has_more** (resource result)

Finds whether more rows are available from the given result set.

Lista de parámetros

result

The SQLite result resource.

Valores retornados

Returns **TRUE** if there are more rows available from the *result* handle, or **FALSE** otherwise.

Ver también

sqlite_num_rows()
sqlite_changes()

sqlite_has_prev

(PHP 5)

sqlite_has_prev

(no version information, might be only in CVS)

SQLiteResult->hasPrev -- Returns whether or not a previous row is available

Descripción

bool **sqlite_has_prev** (resource result)

Object oriented style (method):

class **SQLiteResult** {

bool **hasPrev** (void)

}

Find whether there are more previous rows from the given result handle.

Lista de parámetros

result

The SQLite result resource. This parameter is not required when using the object-oriented method.

Nota: Esta funcion no puede usarse con resultados que no se encuentren almacenados en un almacenador intermedio (buffer).

Valores retornados

Returns **`TRUE`** if there are more previous rows available from the *result* handle, or **`FALSE`** otherwise.

Ver también

sqlite_prev()
sqlite_has_more()
sqlite_num_rows()

sqlite_key

(no version information, might be only in CVS)

sqlite_key

(no version information, might be only in CVS)

SQLiteResult->key -- Returns the current row index

Descripción

int **sqlite_key** (resource result)

Object oriented style (method):

class **SQLiteResult** {

int **key** (void)

}

sqlite_key() returns the current row index of the buffered result set *result*.

Lista de parámetros

result

The SQLite result resource. This parameter is not required when using the object-oriented method.

Nota: Esta funcion no puede usarse con resultados que no se encuentren almacenados en un almacenador intermedio (buffer).

Valores retornados

Returns the current row index of the buffered result set *result*.

Registro de cambios

Versión	Descripción
5.0.4	Prior to PHP 5.0.4, **sqlite_key()** was only able to be called as a method on a SQLiteResult object, not procedurally.

Ver también

sqlite_next()
sqlite_current()
sqlite_rewind()

sqlite_last_error

(PHP 5)

sqlite_last_error

(no version information, might be only in CVS)

SQLiteDatabase->lastError -- Returns the error code of the last error for a database

Descripción

int **sqlite_last_error** (resource dbhandle)

Object oriented style (method):

class **SQLiteDatabase** {

int **lastError** (void)

}

Returns the error code from the last operation performed on *dbhandle*, the database handle. A human readable description of the error code can be retrieved using **sqlite_error_string()**.

Lista de parámetros

dbhandle

The SQLite Database resource; returned from **sqlite_open ()** when used procedurally. This parameter is not required when using the object-oriented method.

Ver también

sqlite_error_string()

sqlite_last_insert_rowid

(PHP 5)

sqlite_last_insert_rowid

(no version information, might be only in CVS)

SQLiteDatabase->lastInsertRowid -- Returns the rowid of the most recently inserted row

Descripción

int **sqlite_last_insert_rowid** (resource dbhandle)

Object oriented style (method):

class **SQLiteDatabase** {

int **lastInsertRowid** (void)

}

Returns the rowid of the row that was most recently inserted into the database *dbhandle*, if it was created as an auto-increment field.

Sugerencia: You can create auto-increment fields in SQLite by declaring them as *INTEGER PRIMARY KEY* in your table schema.

Lista de parámetros

dbhandle

The SQLite Database resource; returned from **sqlite_open ()** when used procedurally. This parameter is not required when using the object-oriented method.

sqlite_libencoding

(PHP 5)

sqlite_libencoding -- Returns the encoding of the linked SQLite library

Descripción

string **sqlite_libencoding** (void)

The SQLite library may be compiled in either ISO-8859-1 or UTF-8 compatible modes. This function allows you to determine which encoding scheme is used by your version of the library.

Aviso
The default PHP distribution builds libsqlite in ISO-8859-1 encoding mode. However, this is a misnomer; rather than handling ISO-8859-1, it operates according to your current locale settings for string comparisons and sort ordering. So, rather than ISO-8859-1, you should think of it as being '8-bit' instead.

When compiled with UTF-8 support, sqlite handles encoding and decoding of UTF-8 multi-byte character sequences, but does not yet do a complete job when working with the data (no normalization is performed for example), and some comparison operations may still not be carried out correctly.

Aviso
It is not recommended that you use PHP in a web-server configuration with a version of the SQLite library compiled with UTF-8 support, since libsqlite will abort the process if it detects a problem with the UTF-8 encoding.

Ver también

sqlite_lib_version()

sqlite_libversion

(PHP 5)

sqlite_libversion -- Returns the version of the linked SQLite library

Description

string **sqlite_libversion** (void)

Returns the version of the linked SQLite library.

Ver también

sqlite_libencoding()

sqlite_next

(PHP 5)

sqlite_next

(no version information, might be only in CVS)

SQLiteResult->next

(no version information, might be only in CVS)

SQLiteUnbuffered->next -- Seek to the next row number

Descripción

bool **sqlite_next** (resource result)

Object oriented style (method):

class **SQLiteResult** {

bool **next** (void)

}class **SQLiteUnbuffered** {

bool **next** (void)

}

sqlite_next() advances the result handle *result* to the next row.

Lista de parámetros

result

The SQLite result resource. This parameter is not required when using the object-oriented method.

Valores retornados

Returns **`TRUE`** on success, or **`FALSE`** if there are no more rows.

Ver también

sqlite_seek()
sqlite_current()
sqlite_rewind()

sqlite_num_fields

(PHP 5)

sqlite_num_fields

(no version information, might be only in CVS)

SQLiteResult->numFields

(no version information, might be only in CVS)

SQLiteUnbuffered->numFields -- Returns the number of fields in a result set

Descripción

int **sqlite_num_fields** (resource result)

Object oriented style (method):

class **SQLiteResult** {

int **numFields** (void)

}class **SQLiteUnbuffered** {

int **numFields** (void)

}

Returns the number of fields in the *result* set.

Lista de parámetros

result

The SQLite result resource. This parameter is not required when using the object-oriented method.

Ver también

sqlite_changes()
sqlite_num_rows()

sqlite_num_rows

(PHP 5)

sqlite_num_rows

(no version information, might be only in CVS)

SQLiteResult->numRows -- Returns the number of rows in a buffered result set

Description

int **sqlite_num_rows** (resource result)

Object oriented style (method):

class **SQLiteResult** {

int **numRows** (void)

}

Returns the number of rows in the buffered *result* set.

Lista de parámetros

result

The SQLite result resource. This parameter is not required when using the object-oriented method.

Nota: Esta funcion no puede usarse con resultados que no se encuentren almacenados en un almacenador intermedio (buffer).

Ejemplos

Ejemplo 1. Procedural example

```
<?php
$db = sqlite_open('mysqlitedb');
$result = sqlite_query($db, "SELECT * FROM mytable WHERE name='John Doe'");
$rows = sqlite_num_rows($result);

echo "Number of rows: $rows";
?>
```

Ejemplo 2. Object-oriented example

```
<?php
$db = new SQLiteDatabase('mysqlitedb');
$result = $db->query("SELECT * FROM mytable WHERE name='John Doe'");
$rows = $result->numRows();

echo "Number of rows: $rows";
?>
```

Ver también

sqlite_changes()
sqlite_query()
sqlite_num_fields()

sqlite_open

(PHP 5)

sqlite_open -- Opens a SQLite database and create the database if it does not exist

Descripción

resource **sqlite_open** (string filename [, int mode [, string &error_message]])

Object oriented style (constructor):

class **SQLiteDatabase** {

__construct (string filename [, int mode [, string &error_message]])

}

Opens a SQLite database or creates the database if it does not exist.

Lista de parámetros

filename

The filename of the SQLite database. If the file does not exist, SQLite will attempt to create it. PHP must have write permissions to the file if data is inserted, the database schema is modified or to create the database if it does not exist.

mode

The mode of the file. Intended to be used to open the database in read-only mode. Presently, this parameter is ignored by the sqlite library. The default value for mode is the octal value *0666* and this is the recommended value.

error_message

Passed by reference and is set to hold a descriptive error message explaining why the database could not be opened if there was an error.

Valores retornados

Returns a resource (database handle) on success, **FALSE** on error.

Ejemplos

Ejemplo 1. sqlite_open() example

```
<?php
if ($db = sqlite_open('mysqlitedb', 0666, $sqliteerror)) {
    sqlite_query($db, 'CREATE TABLE foo (bar varchar(10))');
    sqlite_query($db, "INSERT INTO foo VALUES ('fnord')");
    $result = sqlite_query($db, 'select bar from foo');
    var_dump(sqlite_fetch_array($result));
} else {
    die($sqliteerror);
}
?>
```

Notes

> **Sugerencia:** On Unix platforms, SQLite is sensitive to scripts that use the fork() system call. If you do have such a script, it is recommended that you close the handle prior to forking and then re-open it in the child and/or parent. For more information on this issue, see The C language interface to the SQLite library in the section entitled *Multi-Threading And SQLite*.
>
> **Sugerencia:** It is not recommended to work with SQLite databases mounted on NFS partitions. Since NFS is notoriously bad when it comes to locking you may find that you cannot even open the database at all, and if it succeeds, the locking behaviour may be undefined.
>
> **Nota:** Starting with SQLite library version 2.8.2, you can specify *:memory:* as the *filename* to create a database that lives only in the memory of the computer. This is useful mostly for temporary processing, as the in-memory database will be destroyed when the process ends. It can also be useful when coupled with the *ATTACH DATABASE* SQL statement to load other databases and move and query data between them.
>
> **Nota:** SQLite is safe mode and open_basedir aware.

Ver también

sqlite_popen()
sqlite_close()
sqlite_factory()

sqlite_popen

(PHP 5)

sqlite_popen -- Opens a persistent handle to an SQLite database and create the database if it does not exist

Descripción

resource **sqlite_popen** (string filename [, int mode [, string &error_message]])

This function behaves identically to **sqlite_open()** except that is uses the persistent resource mechanism of PHP. For information about the meaning of the parameters, read the **sqlite_open()** manual page.

sqlite_popen() will first check to see if a persistent handle has already been opened for the given *filename*. If it finds one, it returns that handle to your script, otherwise it opens a fresh handle to the database.

The benefit of this approach is that you don't incur the performance cost of re-reading the database and index schema on each page hit served by persistent web server SAPI's (any SAPI except for regular CGI or CLI).

> **Nota:** If you use persistent handles and have the database updated by a background process (perhaps via a crontab), and that process re-creates the database by overwriting it (either by unlinking and rebuilding, or moving the updated version to replace the current version), you may experience undefined behaviour when a persistent handle on the old version of the database is recycled.
>
> To avoid this situation, have your background processes open the same database file and perform their updates in a transaction.

Lista de parámetros

filename

> The filename of the SQLite database. If the file does not exist, SQLite will attempt to create it. PHP must have write permissions to the file if data is inserted, the database schema is modified or to create the database if it does not exist.

mode

> The mode of the file. Intended to be used to open the database in read-only mode. Presently, this parameter is ignored by the sqlite library. The default value for mode is the octal value *0666* and this is the recommended value.

error_message

> Passed by reference and is set to hold a descriptive error message explaining why the database could not be opened if there was an error.

Valores retornados

Returns a resource (database handle) on success, **`FALSE`** on error.

Ver también

sqlite_open()
sqlite_close()
sqlite_factory()

sqlite_prev

(PHP 5)

sqlite_prev

(no version information, might be only in CVS)

SQLiteResult->prev -- Seek to the previous row number of a result set

Descripción

bool **sqlite_prev** (resource result)

Object oriented style (method):

class **SQLiteResult** {

bool **prev** (void)

}

sqlite_prev() seeks back the *result* handle to the previous row.

Lista de parámetros

result

The SQLite result resource. This parameter is not required when using the object-oriented method.

Nota: Esta funcion no puede usarse con resultados que no se encuentren almacenados en un almacenador intermedio (buffer).

Valores retornados

Returns **`TRUE`** on success, or **`FALSE`** if there are no more previous rows.

Ver también

sqlite_has_prev()
sqlite_rewind()
sqlite_next()

sqlite_query

(PHP 5)

sqlite_query

(no version information, might be only in CVS)

SQLiteDatabase->query -- Executes a query against a given database and returns a result handle

Descripción

resource **sqlite_query** (resource dbhandle, string query)

resource **sqlite_query** (string query, resource dbhandle)

Object oriented style (method):

class **SQLiteDatabase** {

SQLiteResult **query** (string query)

}

Executes an SQL statement given by the *query* against a given database handle.

Lista de parámetros

query

The query to be executed.

dbhandle

The SQLite Database resource; returned from **sqlite_open ()** when used procedurally. This parameter is not required when using the object-oriented method.

Nota: Two alternative syntaxes are supported for compatibility with other database extensions (such as MySQL). The preferred form is the first, where the *dbhandle* parameter is the first parameter to the function.

Valores retornados

This function will return a result handle or **`FALSE`** on failure. For queries that return rows, the result handle can then be used with functions such as **sqlite_fetch_array()** and **sqlite_seek()**.

Regardless of the query type, this function will return **`FALSE`** if the query failed.

sqlite_query() returns a buffered, seekable result handle. This is useful for reasonably small queries where you need to be able to randomly access the rows. Buffered result handles will allocate memory to hold the entire result and will not return until it has been fetched. If you only need sequential access to the data, it is recommended that you use the much higher performance **sqlite_unbuffered_query()** instead.

Notes

Aviso

SQLite *will* execute multiple queries separated by semicolons, so you can use it to execute a batch of SQL that you have loaded from a file or have embedded in a script. However, this works only when the result of the function is not used - if it is used, only the first SQL statement would be executed. Function **sqlite_exec()** will always execute multiple SQL statements.

When executing multiple queries, the return value of this function will be **FALSE** if there was an error, but undefined otherwise (it might be **TRUE** for success or it might return a result handle).

Ver también

sqlite_unbuffered_query()
sqlite_array_query()

sqlite_rewind

(PHP 5)

sqlite_rewind

(no version information, might be only in CVS)

SQLiteResult->rewind -- Seek to the first row number

Descripción

bool **sqlite_rewind** (resource result)

Object oriented style (method):

class **SQLiteResult** {

bool **rewind** (void)

}

sqlite_rewind() seeks back to the first row in the given result set.

Lista de parámetros

result

The SQLite result resource. This parameter is not required when using the object-oriented method.

Nota: Esta funcion no puede usarse con resultados que no se encuentren almacenados en un almacenador intermedio (buffer).

Valores retornados

Returns **FALSE** if there are no rows in the result set, **TRUE** otherwise.

Ver también

sqlite_next()
sqlite_current()
sqlite_seek()

sqlite_seek

(PHP 5)

sqlite_seek

(no version information, might be only in CVS)

SQLiteResult->seek -- Seek to a particular row number of a buffered result set

Descripción

bool **sqlite_seek** (resource result, int rownum)

Object oriented style (method):

class **SQLiteResult** {

bool **seek** (int rownum)

}

sqlite_seek() seeks to the row given by the parameter *rownum*.

Lista de parámetros

result

The SQLite result resource. This parameter is not required when using the object-oriented method.

Nota: Esta funcion no puede usarse con resultados que no se encuentren almacenados en un almacenador intermedio (buffer).

rownum

The ordinal row number to seek to. The row number is zero-based (0 is the first row).

Nota: Esta funcion no puede usarse con resultados que no se encuentren almacenados en un almacenador intermedio (buffer).

Valores retornados

Returns **`FALSE`** if the row does not exist, **`TRUE`** otherwise.

Ver también

sqlite_next()
sqlite_current()
sqlite_rewind()

sqlite_single_query

(PHP 5)

sqlite_single_query

(no version information, might be only in CVS)

SQLiteDatabase->singleQuery -- Executes a query and returns either an array for one single column or the value of the first row

Descripción

mixed **sqlite_single_query** (resource db, string query [, bool first_row_only [, bool decode_binary]])

Object oriented style (method):

class **SQLiteDatabase** {

mixed **singleQuery** (string query [, bool first_row_only [, bool decode_binary]])

}

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

sqlite_udf_decode_binary

(PHP 5)

sqlite_udf_decode_binary -- Decode binary data passed as parameters to an UDF

Descripción

string **sqlite_udf_decode_binary** (string data)

sqlite_udf_decode_binary() decodes the binary encoding that was applied to the parameter by either **sqlite_udf_encode_binary()** or **sqlite_escape_string()**.

You must call this function on parameters passed to your UDF if you need them to handle binary data, as the binary encoding employed by PHP will obscure the content and of the parameter in its

natural, non-coded form.

PHP does not perform this encode/decode operation automatically as it would severely impact performance if it did.

Ejemplos

Ejemplo 1. binary-safe max_length aggregation function example

```
<?php
$data = array(
   'one',
   'two',
   'three',
   'four',
   'five',
   'six',
   'seven',
   'eight',
   'nine',
   'ten',
   );
$db = sqlite_open(':memory:');
sqlite_query($db, "CREATE TABLE strings(a)");
foreach ($data as $str) {
    $str = sqlite_escape_string($str);
    sqlite_query($db, "INSERT INTO strings VALUES ('$str')");
}

function max_len_step(&$context, $string)
{
    $string = sqlite_udf_decode_binary($string);
    if (strlen($string) > $context) {
        $context = strlen($string);
    }
}

function max_len_finalize(&$context)
{
    return $context;
}

sqlite_create_aggregate($db, 'max_len', 'max_len_step', 'max_len_finalize');

var_dump(sqlite_array_query($db, 'SELECT max_len(a) from strings'));

?>
```

Ver también

sqlite_udf_encode_binary()
sqlite_create_function()
sqlite_create_aggregate()

sqlite_udf_encode_binary

(PHP 5)

sqlite_udf_encode_binary -- Encode binary data before returning it from an UDF

Descripción

string **sqlite_udf_encode_binary** (string data)

sqlite_udf_encode_binary() applies a binary encoding to the *data* so that it can be safely returned from queries (since the underlying libsqlite API is not binary safe).

If there is a chance that your data might be binary unsafe (e.g.: it contains a NUL byte in the middle rather than at the end, or if it has and *0x01* byte as the first character) then you must call this function to encode the return value from your UDF.

PHP does not perform this encode/decode operation automatically as it would severely impact performance if it did.

> **Nota:** Do not use **sqlite_escape_string()** to quote strings returned from UDF's as it will lead to double-quoting of the data. Use **sqlite_udf_encode_binary()** instead!

Ver también

sqlite_udf_decode_binary()
sqlite_escape_string()
sqlite_create_function()
sqlite_create_aggregate()

sqlite_unbuffered_query

(PHP 5)

sqlite_unbuffered_query

(no version information, might be only in CVS)

SQLiteDatabase->unbufferedQuery -- Execute a query that does not prefetch and buffer all data

Descripción

resource **sqlite_unbuffered_query** (resource dbhandle, string query)

resource **sqlite_unbuffered_query** (string query, resource dbhandle)

Object oriented style (method):

class **SQLiteDatabase** {

SQLiteUnbuffered **unbufferedQuery** (string query)

}

sqlite_unbuffered_query() is identical to **sqlite_query()** except that the result that is returned is a sequential forward-only result set that can only be used to read each row, one after the other.

This function is ideal for generating things such as HTML tables where you only need to process one row at a time and don't need to randomly access the row data.

> **Nota:** Functions such as **sqlite_seek()**, **sqlite_rewind()**, **sqlite_next()**, **sqlite_current()**, and **sqlite_num_rows()** do not work on result handles returned from **sqlite_unbuffered_query()**.

Lista de parámetros

query

> The query to be executed.

dbhandle

> The SQLite Database resource; returned from **sqlite_open ()** when used procedurally. This parameter is not required when using the object-oriented method.
>
> **Nota:** Two alternative syntaxes are supported for compatibility with other database extensions (such as MySQL). The preferred form is the first, where the *dbhandle* parameter is the first parameter to the function.

Valores retornados

Returns a result handle or **`FALSE`** on failure.

sqlite_unbuffered_query() returns a sequential forward-only result set that can only be used to read each row, one after the other.

Ver también

sqlite_query()

sqlite_valid

(PHP 5)

sqlite_valid

(no version information, might be only in CVS)

SQLiteResult->valid

(no version information, might be only in CVS)

SQLiteUnbuffered->valid -- Returns whether more rows are available

Descripción

bool **sqlite_valid** (resource result)

Object oriented style (method):

class **SQLiteResult** {

bool **valid** (void)

}class **SQLiteUnbuffered** {

bool **valid** (void)

}

Finds whether more rows are available from the given result handle.

Lista de parámetros

result

The SQLite result resource. This parameter is not required when using the object-oriented method.

Nota: Esta funcion no puede usarse con resultados que no se encuentren almacenados en un almacenador intermedio (buffer).

Valores retornados

Returns **`TRUE`** if there are more rows available from the *result* handle, or **`FALSE`** otherwise.

Ver también

sqlite_num_rows()
sqlite_changes()

CXIX. Secure Shell2 Functions

Introducción

Bindings to the libssh2 library which provide access to resources (shell, remote exec, tunneling, file transfer) on a remote machine using a secure cryptographic transport.

Instalación

Mas informacion sobre nuevos lanzamientos, descargas ficheros de fuentes, informacion sobre los responsables asi como un 'CHANGELOG', se puede encontrar aqui: http://pecl.php.net/package/ssh2.

You will also need version 0.4 or greater of the libssh2 library (possibly higher, see release notes).

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

SSH2_FINGERPRINT_MD5 (integer)

Flag to ssh2_fingerprint() requesting hostkey fingerprint as an MD5 hash.

SSH2_FINGERPRINT_SHA1 (integer)

Flag to ssh2_fingerprint() requesting hostkey fingerprint as an SHA1 hash.

SSH2_FINGERPRINT_HEX (integer)

Flag to ssh2_fingerprint() requesting hostkey fingerprint as a string of hexits.

SSH2_FINGERPRINT_RAW (integer)

Flag to ssh2_fingerprint() requesting hostkey fingerprint as a raw string of 8-bit characters.

SSH2_TERM_UNIT_CHARS (integer)

Flag to ssh2_shell() specifying that *width* and *height* are provided as character sizes.

SSH2_TERM_UNIT_PIXELS (integer)

Flag to ssh2_shell() specifying that *width* and *height* are provided in pixel units.

SSH2_DEFAULT_TERM_WIDTH (integer)

Default terminal width requested by ssh2_shell().

SSH2_DEFAULT_TERM_HEIGHT (integer)

Default terminal height requested by ssh2_shell().

SSH2_DEFAULT_TERM_UNIT (integer)

Default terminal units requested by ssh2_shell().

SSH2_STREAM_STDIO (integer)

Flag to ssh2_fetch_stream() requesting STDIO subchannel.

SSH2_STREAM_STDERR (integer)

Flag to ssh2_fetch_stream() requesting STDERR subchannel.

SSH2_DEFAULT_TERMINAL (string)

Default terminal type (e.g. vt102, ansi, xterm, vanilla) requested by **ssh2_shell()**.

Tabla de contenidos

ssh2_auth_hostbased_file

(no version information, might be only in CVS)

ssh2_auth_hostbased_file -- Authenticate using a public hostkey

Description

bool **ssh2_auth_hostbased_file** (resource session, string username, string hostname, string pubkeyfile, string privkeyfile [, string passphrase [, string local_username]])

Authenticate using a public hostkey read from a file. If *privkeyfile* is encrypted (which it should be), the passphrase must be provided. If *local_username* is omitted, then the value for *username* will be used for it.

Ejemplo 1. Authentication using a public hostkey

```
<?php
$connection = ssh2_connect('shell.example.com', 22, array('hostkey'=>'ssh-rsa'));

if (ssh2_auth_hostbased_file($connection, 'remoteusername', 'myhost.example.com',
                             '/usr/local/etc/hostkey_rsa.pub',
                             '/usr/local/etc/hostkey_rsa', 'secret',
                             'localusername')) {
  echo "Public Key Hostbased Authentication Successful\n";
} else {
  die('Public Key Hostbased Authentication Failed');
}
?>
```

Nota: ssh2_auth_hostbased_file() requires libssh2 >= 0.7 and PHP/SSH2 >= 0.7

ssh2_auth_none

(no version information, might be only in CVS)

ssh2_auth_none -- Authenticate as "none"

Description

array **ssh2_auth_none** (resource session, string username)

Attempt "none" authentication which usually will (and should) fail. As part of this failure, servers will return a list of accepted authentication methods. If the server does accept "none" as an authentication method for *username*, this function will simply return **TRUE**.

Ejemplo 1. Using ssh2_auth_none() to retreive a list of authentication methods.

```
<?php
$connection = ssh2_connect('shell.example.com', 22);

$auth_methods = ssh2_auth_none($connection, 'user');

if (in_array('password', $auth_methods)) {
  echo "Server supports password based authentication\n";
}
?>
```

ssh2_auth_password

(no version information, might be only in CVS)

ssh2_auth_password -- Authenticate over SSH using a plain password

Description

bool **ssh2_auth_password** (resource session, string username, string password)

Authenticate over SSH using a plain password

Ejemplo 1. Authenticating with a password

```
<?php
$connection = ssh2_connect('shell.example.com', 22);

if (ssh2_auth_password($connection, 'username', 'secret')) {
  echo "Authentication Successful!\n";
} else {
  die('Authentication Failed...');
}
?>
```

ssh2_auth_pubkey_file

(no version information, might be only in CVS)

ssh2_auth_pubkey_file -- Authenticate using a public key

Description

bool **ssh2_auth_pubkey_file** (resource session, string username, string pubkeyfile, string privkeyfile [, string passphrase])

Authenticate using a public key read from a file. If *privkeyfile* is encrypted (which it should be), the passphrase must be provided.

Ejemplo 1. Authentication using a public key

```
<?php
$connection = ssh2_connect('shell.example.com', 22, array('hostkey'=>'ssh-rsa'));

if (ssh2_auth_pubkey_file($connection, 'username',
                          '/home/username/.ssh/id_rsa.pub',
                          '/home/username/.ssh/id_rsa', 'secret')) {
  echo "Public Key Authentication Successful\n";
} else {
  die('Public Key Authentication Failed');
}
?>
```

ssh2_connect

(no version information, might be only in CVS)

ssh2_connect -- Connect to an SSH server

Description

resource **ssh2_connect** (string host [, int port [, array methods [, array callbacks]]])

Establish a connection to a remote SSH server and return a resource on success, **FALSE** on error.

methods may be an associative array with up to four parameters as described below.

Tabla 1. *methods* may be an associative array with any or all of the following parameters.

Index	Meaning	Supported Values*
kex	List of key exchange methods to advertise, coma separated in order of preference.	*diffie-hellman-group1-sha1*, *diffie-hellman-group14-sha1*, and *diffie-hellman-group-exchange-sha1*
hostkey	List of hostkey methods to advertise, come separated in order of preference.	*ssh-rsa* and *ssh-dss*
client_to_server	Associative array containing crypt, compression, and message authentication code (MAC) method preferences for messages sent from client to server.	

Index	Meaning	Supported Values*
server_to_client	Associative array containing crypt, compression, and message authentication code (MAC) method preferences for messages sent from client to server.	

* - Supported Values are dependent on methods supported by underlying library. See libssh2 documentation for additional information.

Tabla 2. *client_to_server* and *server_to_client* may be an associative array with any or all of the following parameters.

Index	Meaning	Supported Values*
crypt	List of crypto methods to advertise, coma separated in order of preference.	*rijndael-cbc@lysator.liu.se*, *aes256-cbc*, *aes192-cbc*, *aes128-cbc*, *3des-cbc*, *blowfish-cbc*, *cast128-cbc*, *arcfour*, and *none***
comp	List of compression methods to advertise, coma separated in order of preference.	*zlib* and *none*
mac	List of MAC methods to advertise, come separated in order of preference.	*hmac-sha1*, *hmac-sha1-96*, *hmac-ripemd160*, *hmac-ripemd160@openssh.com*, and *none***

Crypt and MAC method "*none*": For security reasons, *none* is disabled by the underlying libssh2 library unless explicitly enabled during build time by using the appropriate ./configure options. See documentation for the underlying library for more information.

Tabla 3. *callbackss* may be an associative array with any or all of the following parameters.

Index	Meaning	Prototype
ignore	Name of function to call when an **SSH2_MSG_IGNORE** packet is received	void ignore_cb ($message)
debug	Name of function to call when an **SSH2_MSG_DEBUG** packet is received	void debug_cb ($message, $language, $always_display)
macerror	Name of function to call when a packet is received but the message authentication code failed. If the callback returns **TRUE**, the mismatch will be ignored, otherwise the connection will be terminated.	bool macerror_cb ($packet)
disconnect	Name of function to call when an **SSH2_MSG_DISCONNECT** packet is received	void disconnect_cb ($reason, $message, $language)

Ejemplo 1. Open a connection forcing 3des-cbc when sending packets, any strength aes cipher when receiving packets, no compression in either direction, and Group1 key exchange.

```
<?php
/* Notify the user if the server terminates the connection */
function my_ssh_disconnect($reason, $message, $language) {
  printf("Server disconnected with reason code [%d] and message: %s\n",
         $reason, $message);
}

$methods = array(
  'kex' => 'diffie-hellman-group1-sha1',
  'client_to_server' => array(
    'crypt' => '3des-cbc',
    'comp' => 'none'),
  'server_to_client' => array(
    'crypt' => 'aes256-cbc,aes192-cbc,aes128-cbc',
    'comp' => 'none'));

$callbacks = array('disconnect' => 'my_ssh_disconnect');

$connection = ssh2_connect('shell.example.com', 22, $methods, $callbacks);
if (!$connection) die('Connection failed');
?>
```

Once connected, the client should verify the server's hostkey using **ssh2_fingerprint()**, then authenticate using either password or public key.

See Also: **ssh2_fingerprint()**, **ssh2_auth_none()**, **ssh2_auth_password()**, and **ssh2_auth_pubkey_file()**

ssh2_exec

(no version information, might be only in CVS)

ssh2_exec -- Execute a command on a remote server

Description

stream **ssh2_exec** (resource session, string command [, array env])

Execute a command at the remote end and allocate a channel for it. Returns a stream on success or **FALSE** on failure.

Ejemplo 1. Executing a command

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');

$stream = ssh2_exec($connection, '/usr/local/bin/php -i');
?>
```

See Also: **ssh2_connect()**, **ssh2_shell()**, and **ssh2_tunnel()**

ssh2_fetch_stream

(no version information, might be only in CVS)

ssh2_fetch_stream -- Fetch an extended data stream

Description

stream **ssh2_fetch_stream** (stream channel, int streamid)

Fetches an alternate substream associated with an SSH2 channel stream identified by *streamid*. The SSH2 protocol currently defines only one substream, STDERR, which has a substream ID of **SSH2_STREAM_STDERR** (defined as 1).

Ejemplo 1. Opening a shell and retrieving the stderr stream associated with it.

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');

$stdio_stream = ssh2_shell($connection);
$stderr_stream = ssh2_fetch_stream($stdio_stream, SSH2_STREAM_STDERR);
?>
```

See Also: **ssh2_shell()**, **ssh2_exec()**, and **ssh2_connect()**

ssh2_fingerprint

(no version information, might be only in CVS)

ssh2_fingerprint -- Retreive fingerprint of remote server

Description

string **ssh2_fingerprint** (resource session [, int flags])

Returns a server hostkey hash from an active session. Defaults to MD5 fingerprint encoded as ASCII hex values.

flags may be either of **SSH2_FINGERPRINT_MD5** or **SSH2_FINGERPRINT_SHA1** logically ORed with **SSH2_FINGERPRINT_HEX** or **SSH2_FINGERPRINT_RAW**. Defaults to **SSH2_FINGERPRINT_MD5** | **SSH2_FINGERPRINT_HEX**.

Ejemplo 1. Checking the fingerprint against a known value

```
<?php
$known_host = '6F89C2F0A719B30CC38ABDF90755F2E4';

$connection = ssh2_connect('shell.example.com', 22);

$fingerprint = ssh2_fingerprint($connection,
               SSH2_FINGERPRINT_MD5 | SSH2_FINGERPRINT_HEX);

if ($fingerprint != $known_host) {
  die("HOSTKEY MISMATCH!\n" .
      "Possible Man-In-The-Middle Attack?");
}
?>
```

ssh2_methods_negotiated

(no version information, might be only in CVS)

ssh2_methods_negotiated -- Return list of negotiated methods

Description

array **ssh2_methods_negotiated** (resource session)

Returns list of negotiated methods.

Ejemplo 1. Determining what methods were negotiated

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
$methods = ssh2_methods_negotiated($connection);

echo "Encryption keys were negotiated using: {$methods['kex']}\n";
echo "Server identified using an {$methods['hostkey']} with ";
echo "fingerprint: " . ssh2_fingerprint($connection) . "\n";

echo "Client to Server packets will use methods:\n";
echo "\tCrypt: {$methods['client_to_server']['crypt']}\n";
echo "\tComp: {$methods['client_to_server']['comp']}\n";
echo "\tMAC: {$methods['client_to_server']['mac']}\n";

echo "Server to Client packets will use methods:\n";
echo "\tCrypt: {$methods['server_to_client']['crypt']}\n";
echo "\tComp: {$methods['server_to_client']['comp']}\n";
echo "\tMAC: {$methods['server_to_client']['mac']}\n";

?>
```

See Also: **ssh2_connect()**

ssh2_scp_recv

(no version information, might be only in CVS)

ssh2_scp_recv -- Request a file via SCP

Description

bool **ssh2_scp_recv** (resource session, string remote_file, string local_file)

Copy a file from the remote server to the local filesystem using the SCP protocol.

Ejemplo 1. Downloading a file via SCP

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');

ssh2_scp_recv($connection, '/remote/filename', '/local/filename');
?>
```

See Also: **ssh2_scp_send()**, and **copy()**

ssh2_scp_send

(no version information, might be only in CVS)

ssh2_scp_send -- Send a file via SCP

Description

stream **ssh2_scp_send** (resource session, string local_file, string remote_file [, int create_mode])

Copy a file from the local filesystem to the remote server using the SCP protocol. The file will be created with the mode specified by *create_mode*.

Ejemplo 1. Uploading a file via SCP

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');

ssh2_scp_send($connection, '/local/filename', '/remote/filename', 0644);
?>
```

See Also: **ssh2_scp_recv()**, and **copy()**

ssh2_sftp_lstat

(no version information, might be only in CVS)

ssh2_sftp_lstat -- Stat a symbolic link

Description

array **ssh2_sftp_lstat** (resource sftp, string path)

Stats a symbolic link on the remote filesystem *without* following the link. This function is similar to using the **lstat()** function with the ssh2.sftp:// wrapper in PHP5 and returns the same values. See the documentation for **stat()** for details on the values which may be returned.

Ejemplo 1. Stating a symbolic link via SFTP

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');

$sftp = ssh2_sftp($connection);
$statinfo = ssh2_lstat($sftp, '/path/to/symlink');

$filesize = $statinfo['size'];
$group = $statinfo['gid'];
$owner = $statinfo['uid'];
$atime = $statinfo['atime'];
$mtime = $statinfo['mtime'];
$mode = $statinfo['mode'];
?>
```

See Also: **ssh2_sftp_stat()**, **lstat()**, and **stat()**

ssh2_sftp_mkdir

(no version information, might be only in CVS)

ssh2_sftp_mkdir -- Create a directory

Description

bool **ssh2_sftp_mkdir** (resource sftp, string dirname [, int mode [, bool recursive]])

Creates a directory on the remote file server with permissions set to *mode*. If *recursive* is **TRUE** any parent directories required for *dirname* will be automatically created as well. This function is similar to using **mkdir()** with the ssh2.sftp:// wrapper.

Ejemplo 1. Creating a directory on a remote server

```
<?php
$connection = ssh2_connet('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');
$sftp = ssh2_sftp($connection);

ssh2_sftp_mkdir($sftp, '/home/username/newdir');
/* Or:  mkdir("ssh2.sftp://$sftp/home/username/newdir"); */
?>
```

See Also: **mkdir()**, and **ssh2_sftp_rmdir()**

ssh2_sftp_readlink

(no version information, might be only in CVS)

ssh2_sftp_readlink -- Return the target of a symbolic link

Description

string **ssh2_sftp_readlink** (resource sftp, string link)

Returns the target of a symbolic link.

Ejemplo 1. Reading a symbolic link

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');
$sftp = ssh2_sftp($connection);

$target = ssh2_sftp_readlink($sftp, '/tmp/mysql.sock');
/* $target is now (e.g.): '/var/run/mysql.sock' */
?>
```

See Also: **readlink()**, and **ssh2_sftp_symlink()**

ssh2_sftp_realpath

(no version information, might be only in CVS)

ssh2_sftp_realpath -- Resolve the realpath of a provided path string

Description

string **ssh2_sftp_realpath** (resource sftp, string filename)

Translates *filename* into the effective real path on the remote filesystem.

Ejemplo 1. Resolving a pathname

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');
$sftp = ssh2_sftp($connection);

$realpath = ssh2_sftp_realpath($sftp, '/home/username/../../../..//./usr/../etc/passwd'
/* $realpath is now: '/etc/passwd' */
?>
```

See Also: **realpath()**, and **ssh2_sftp_symlink()** **ssh2_sftp_readlink()**

ssh2_sftp_rename

(no version information, might be only in CVS)

ssh2_sftp_rename -- Rename a remote file

Description

bool **ssh2_sftp_rename** (resource sftp, string from, string to)

Renames a file on the remote filesystem.

Ejemplo 1. Renaming a file via sftp

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');
$sftp = ssh2_sftp($connection);

ssh2_sftp_rename($sftp, '/home/username/oldname', '/home/username/newname');
?>
```

See Also: **rename()**

ssh2_sftp_rmdir

(no version information, might be only in CVS)

ssh2_sftp_rmdir -- Remove a directory

Description

bool **ssh2_sftp_rmdir** (resource sftp, string dirname)

Removes a directory from the remote file server. This function is similar to using **rmdir()** with the ssh2.sftp:// wrapper.

Ejemplo 1. Removing a directory on a remote server

```
<?php
$connection = ssh2_connet('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');
$sftp = ssh2_sftp($connection);

ssh2_sftp_rmdir($sftp, '/home/username/deltodel');
/* Or:  rmdir("ssh2.sftp://$sftp/home/username/dirtodel"); */
?>
```

See Also: **rmdir()**, and **ssh2_sftp_mkdir()**

ssh2_sftp_stat

(no version information, might be only in CVS)

ssh2_sftp_stat -- Stat a file on a remote filesystem

Description

array **ssh2_sftp_stat** (resource sftp, string path)

Stats a file on the remote filesystem following any symbolic links. This function is similar to using the **stat()** function with the ssh2.sftp:// wrapper in PHP5 and returns the same values. See the documentation for **stat()** for details on the values which may be returned.

Ejemplo 1. Stating a file via SFTP

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');

$sftp = ssh2_sftp($connection);
$statinfo = ssh2_stat($sftp, '/path/to/symlink');

$filesize = $statinfo['size'];
$group = $statinfo['gid'];
$owner = $statinfo['uid'];
$atime = $statinfo['atime'];
$mtime = $statinfo['mtime'];
$mode = $statinfo['mode'];
?>
```

See Also: **ssh2_sftp_lstat()**, **lstat()**, and **stat()**

ssh2_sftp_symlink

(no version information, might be only in CVS)

ssh2_sftp_symlink -- Create a symlink

Description

bool **ssh2_sftp_symlink** (resource sftp, string target, string link)

Creates a symbolic link named *link* on the remote filesystem pointing to *target*.

Ejemplo 1. Creating a symbolic link

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');
$sftp = ssh2_sftp($connection);

ssh2_sftp_symlink($sftp, '/var/run/mysql.sock', '/tmp/mysql.sock');
?>
```

See Also: **symlink()**, and **ssh2_sftp_readlink()**

ssh2_sftp_unlink

(no version information, might be only in CVS)

ssh2_sftp_unlink -- Delete a file

Description

bool **ssh2_sftp_unlink** (resource sftp, string filename)

Deletes a file on the remote filesystem

Ejemplo 1. Deleting a file

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');
$sftp = ssh2_sftp($connection);

ssh2_sftp_unlink($sftp, '/home/username/stale_file');
?>
```

See Also: **unlink()**

ssh2_sftp

(no version information, might be only in CVS)

ssh2_sftp -- Initialize SFTP subsystem

Description

resource **ssh2_sftp** (resource session)

Request the SFTP subsystem from an already connected SSH2 server.

This method returns an *SSH2 SFTP* resource for use with all other ssh2_sftp_*() methods and the ssh2.sftp:// fopen wrapper.

Ejemplo 1. Opening a file via SFTP

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');

$sftp = ssh2_sftp($connection);

$stream = fopen("ssh2.sftp://$sftp/path/to/file", 'r');
?>
```

See Also: **ssh2_scp_send()**, and **ssh2_scp_recv()**

ssh2_shell

(no version information, might be only in CVS)

ssh2_shell -- Request an interactive shell

Description

stream **ssh2_shell** (resource session [, string term_type [, array env [, int width [, int height [, int width_height_type]]]]])

Open a shell at the remote end and allocate a stream for it. *term_type* should correspond to one of the entries in the target system's */etc/termcap* file and defaults to *vanilla*. *env* may be passed as an associative array of name/value pairs to set in the target environment.

width, and *height* define the width and height of the virtual terminal allocated for the shell process. *width_height_type* should be one of **SSH2_TERM_UNIT_CHARS** or **SSH2_TERM_UNIT_PIXELS**.

Ejemplo 1. Executing a command

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');

$stream = ssh2_shell($connection, 'vt102', null, 80, 24, SSH2_TERM_UNIT_CHARS);
?>
```

See Also: ssh2_exec(), ssh2_tunnel(), and ssh2_fetch_stream()

ssh2_tunnel

(no version information, might be only in CVS)

ssh2_tunnel -- Open a tunnel through a remote server

Description

stream **ssh2_tunnel** (resource session, string host, int port)

Open a socket stream to an arbitrary host/port by way of the currently connected SSH server.

Ejemplo 1. Opening a tunnel to an arbitrary host

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_pubkey_file($connection, 'username', 'id_dsa.pub', 'id_dsa');

$tunnel = ssh2_tunnel($connection, '10.0.0.101', 12345);
?>
```

See Also: ssh2_connect(), and fsockopen()

CXX. Funciones de Secuencias

Introducción

Las secuencias (streams) fueron introducidas con PHP 4.3.0 como un medio de generalizar el acceso a archivos, recursos de red, compresión de datos, y otras operaciones que comparten un juego común de funciones y usos. En su forma más simple, una *secuencia* es un objeto *recurso* que exhibe

un comportamiento secuenciable. Esto quiere decir, pueden leerse datos desde la secuencia o escribir datos hacia ella en una forma lineal, y puede que sea posible efectuar búsquedas con **fseek()** de ubicaciones arbitrarias dentro de la secuencia.

Una envoltura (*wrapper*) es un código adicional que le dice a la secuencia cómo gestionar los protocolos y codificaciones específicas. Por ejemplo, la envoltura *http* sabe cómo traducir una URL a una petición *HTTP/1.0* por un archivo en un servidor remoto. Existen varias envolturas incluidas con PHP por defecto (Vea Apéndice L), y envolturas adicionales, personalizadas, pueden agregarse ya sea dentro de un script PHP usando **stream_wrapper_register()**, o directamente desde una extensión usando la Referencia de API en Capítulo 63. Dado que cualquier tipo de envoltura puede ser agregada a PHP, no existe un límite impuesto en lo que se puede hacer con ellas. Para consultar la lista de envolturas soportadas actualmente, use **stream_get_wrappers()**.

Una secuencia es referenciada como: *esquema*://*destino*

- *esquema*(cadena) - El nombre de la envoltura a ser usada. Algunos ejemplos: file, http, https, ftp, ftps, compress.zlib, compress.bz2, y php. Vea Apéndice L para consultar una lista de envolturas integradas con PHP. Si no se especifica una envoltura, es usada la envoltura predeterminada de la función (usualmente *file*://).

- *destino* - Depende en la envoltura usada. Para secuencias relacionadas con el sistema de archivos, éste parámetro es por lo general una ruta y un nombre de archivo que apunta al archivo deseado. Para secuencias de red, usualmente consiste de un nombre de host, por lo general con una ruta adicionada al final. Nuevamente, vea Apéndice L para encontrar una descripción de destinos para las secuencias integradas.

Filtros de Secuencia

Un *filtro* es una pieza final de código que puede efectuar operaciones sobre los datos a medida que éstos son leídos desde una secuencia o escritos hacia una. Puede apilarse cualquier cantidad de filtros sobre una secuencia. Pueden definirse filtros personalizados en un script PHP usando **stream_filter_register()** o en una extensión usando la Referencia API de Capítulo 63. Para consultar la lista de filtros registrados actualmente, use **stream_get_filters()**.

Contextos de Secuencia

Un *contexto* es un conjunto de *parámetros* y *opciones* específicas de cada envoltura que modifican o mejoran el comportamiento de una secuencia. Los *contextos* son creados usando **stream_context_create()** y pueden ser pasados a la mayoría de funciones de creación de secuencias relacionadas con el sistema de archivos (esto es, **fopen()**, **file()**, **file_get_contents()**, etc...).

Pueden especificarse *opciones* cuando se hacen llamados a **stream_context_create()**, o más adelante usando **stream_context_set_option()**. Una lista de *opciones* específicas de envoltura puede encontrarse con la lista de envolturas integradas (Vea Apéndice L).

Adicionalmente, pueden definirse *parámetros* en un *contexto* usando **stream_context_set_params()**. Actualmente, el único *parámetro de contexto* soportado por PHP es *notificación*. El valor de éste parámetro debe ser el nombre de una función a ser llamada cuando un evento ocurre sobre una secuencia. La función de notificación llamada durante un evento debe aceptar los siguientes seis parámetros:

void **mi_notificador** (int codigo_notificacion, int severidad, string mensaje, int codigo_mensaje, int bytes_transferidos, int bytes_max)

codigo_mensaje y *severidad* son valores numéricos que corresponden a las constantes **STREAM_NOTIFY_*** listadas más adelante. Si un mensaje descriptivo se encuentra disponible desde la secuencia, *mensaje* y *codigo_mensaje* se popularán con los valores apropiados. El significado de éstos valores depende de la envoltura específica en uso. *bytes_transferidos* y *bytes_max* se popularán cuando sea aplicable.

Instalación

Las secuencias son parte integral de PHP a partir de la versión 4.3.0. No se requiere de ningún paso adicional para habilitarlas.

Clases de Secuencia

Es posible registrar envolturas diseñadas por el usuario mediante **stream_wrapper_register()**, use la definición de clase expuesta en su respectiva página del manual.

La *clase* php_user_filter se encuentra predefinida y es una clase base abstracta para su uso con filtros definidos por el usuario. Vea la página del manual sobre **stream_filter_register()** para más detalles sobre la implementación de filtros definidos por el usuario.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

Constante	Descripción
STREAM_FILTER_READ *	Usada con **stream_filter_append()** y **stream_filter_prepend()** para indicar que el filtro especificado debería ser aplicado únicamente cuando se esté efectuando *lectura*.
STREAM_FILTER_WRITE *	Usada con **stream_filter_append()** y **stream_filter_prepend()** para indicar que el filtro especificado debería ser aplicado únicamente cuando se esté efectuando *escritura*.
STREAM_FILTER_ALL *	Esta constante es ñalente a ***STREAM_FILTER_READ*** | ***STREAM_FILTER_WRITE***.
PSFS_PASS_ON *	*Código de Retorno* que indica que el filtro de espacio de usuario devolvió paquetes en *$salida*.
PSFS_FEED_ME *	*Código de Retorno* que indica que el filtro de espacio de usuario no devolvió paquetes en *$salida* (lo que quiere decir que no hay datos disponibles).
PSFS_ERR_FATAL *	*Código de Retorno* que indica que el filtro de espacio de usuario encontró un error irrecuperable (esto quiere decir, se recibieron datos inválidos).

Constante	Descripción
STREAM_USE_PATH	*Bandera* que indica si la *secuencia* usó la ruta de inclusión.
STREAM_REPORT_ERRORS	*Bandera* que indica si la *envoltura* es responsable de generar errores usando **trigger_error()** durante la apertura de la secuencia. Si esta bandera no está definida, usted no debería generar errores.
STREAM_CLIENT_ASYNC_CONNECT *	Abrir el socket de cliente asincrónicamente. Constante usada con **stream_socket_client()**.
STREAM_CLIENT_PERSISTENT *	El socket de cliente abierto con **stream_socket_client()** debe permanecer persistente entre cargas de la página.
STREAM_SERVER_BIND *	Le dice a una secuencia creada con **stream_socket_server()** que se enlace con el destino especificado. Los sockets de servidor siempre deberían incluir ésta bandera.
STREAM_SERVER_LISTEN *	Le dice a una secuencia creada con **stream_socket_server()** y enlazada usando la bandera **STREAM_SERVER_BIND** que comience a escuchar en el socket. Los transportes orientados a conexiones (como TCP) deben usar esta bandera, de lo contrario el socket de servidor no será habilitado. Usar esta bandera con transportes sin conexión (como UDP) es un error.
STREAM_NOTIFY_RESOLVE *	Una dirección remota requerida para ésta secuencia ha sido resuelta, o la resolución falló. Vea *severidad* para contar con una indicación de lo que ha sucedido.
STREAM_NOTIFY_CONNECT	Se ha establecido una conexión con un recurso externo.
STREAM_NOTIFY_AUTH_REQUIRED	Se requiere de autorización adicional para acceder al recurso especificado. Típicamente se emite con un nivel de *severidad* de **STREAM_NOTIFY_SEVERITY_ERR**.
STREAM_NOTIFY_MIME_TYPE_IS	El *tipo-mime* del recurso ha sido identificado, refiérase a *mensaje* para una descripción del tipo descubierto.
STREAM_NOTIFY_FILE_SIZE_IS	El *tamaño* del recurso ha sido descubierto.
STREAM_NOTIFY_REDIRECTED	El recurso externo ha redireccionado la secuencia a una ubicación alternativa. Refiérase a *mensaje*.
STREAM_NOTIFY_PROGRESS	Indica el progreso actual de una transferencia de secuencia en *bytes_transferidos*, y posiblemente *bytes_max* también.
STREAM_NOTIFY_COMPLETED *	No hay más datos disponibles en la secuencia.
STREAM_NOTIFY_FAILURE	Un error genérico ocurrió en la secuencia, consulte *mensaje* y *codigo_mensaje* para más detalles.
STREAM_NOTIFY_AUTH_RESULT	La autorización se ha completado (con o sin éxito).
STREAM_NOTIFY_SEVERITY_INFO	Notificación normal, no relacionada con errores.
STREAM_NOTIFY_SEVERITY_WARN	Condición de error no-crítico. El procesamiento puede continuar.
STREAM_NOTIFY_SEVERITY_ERR	Un error crítico ha ocurrido. El procesamiento no puede continuar.

Nota: Las constantes marcadas con * se encuentran disponibles únicamente en PHP 5.

Errores de Secuencia

Al igual que con cualquier otra función relacionada con archivos o sockets, una operación sobre una secuencia puede fallar por una variedad de razones normales (esto es: Incapaz de conectarse con el servidor remoto, archivo no encontrado, etc...). Una llamada relacionada con una secuencia puede fallar también debido a que la secuencia deseada no está registrada en el sistema actual. Consulte la matriz devuelta por **stream_get_wrappers()** para ver una lista de secuencias soportadas en su instalación de PHP. Al igual que con la mayoría de funciones internas de PHP, si ocurre un fallo, se generará un mensaje **E_WARNING** que describe la naturaleza del error.

Ejemplos

Ejemplo 1. Uso de file_get_contents() para recuperar datos de múltiples fuentes

```
<?php
/* Leer archivo local desde /home/bar */
$archivo_local = file_get_contents("/home/bar/foo.txt");

/* Identico al ejemplo anterior, indicando explicitamente el esquema FILE */
$archivo_local = file_get_contents("file:///home/bar/foo.txt");

/* Leer un archivo remoto desde www.example.com usando HTTP */
$archivo_http  = file_get_contents("http://www.example.com/foo.txt");

/* Leer un archivo remoto desde www.example.com usando HTTPS */
$archivo_https = file_get_contents("https://www.example.com/foo.txt");

/* Leer un archivo remoto desde ftp.example.com usando FTP */
$archivo_ftp   = file_get_contents("ftp://usuario:contrasenya@ftp.example.com/foo.txt");

/* Leer un archivo remoto desde ftp.example.com usando FTPS */
$archivo_ftps  = file_get_contents("ftps://usuario:contrasenya@ftp.example.com/foo.txt"
?>
```

Ejemplo 2. Realizar una petición POST a un servidor https

```php
<?php
/* Enviar una peticion POST a https://seguro.example.com/formulario.php
 * Incluir elementos de formulario llamados "foo" y "bar" con valores
 * de prueba
 */

$sock = fsockopen("ssl://seguro.example.com", 443, $errno, $errstr, 30);
if (!$sock) die("$errstr ($errno)\n");

$datos = "foo=" . urlencode("Valor para Foo") . "&bar=" . urlencode("Valor para Bar");

fwrite($sock, "POST /formulario.php HTTP/1.0\r\n");
fwrite($sock, "Host: seguro.example.com\r\n");
fwrite($sock, "Content-type: application/x-www-form-urlencoded\r\n");
fwrite($sock, "Content-length: " . strlen($datos) . "\r\n");
fwrite($sock, "Accept: */*\r\n");
fwrite($sock, "\r\n");
fwrite($sock, "$data\r\n");
fwrite($sock, "\r\n");

$cabeceras = "";
while ($cadena = trim(fgets($sock, 4096)))
  $cabeceras .= "$cadena\n";

echo "\n";

$cuerpo = "";
while (!feof($sock))
  $cuerpo .= fgets($sock, 4096);

fclose($sock);
?>
```

Ejemplo 3. Escritura de datos a un archivo comprimido

```php
<?php
/* Crear un archivo comprimido que contenga una cadena arbitraria
 * El archivo puede ser leido de vuelta usando una secuencia
 * compress.zlib o simplemente descomprimido desde la linea de comandos
 * usando 'gzip -d foo-bar.txt.gz'
 */
$da = fopen("compress.zlib://foo-bar.txt.gz", "wb");
if (!$da) die("No fue posible crear el archivo.");

fwrite($da, "Esto es una prueba.\n");

fclose($da);
?>
```

Tabla de contenidos

stream_register_wrapper -- Alias de **stream_wrapper_register()**
stream_select -- Ejecuta el ñalente al llamado de sistema select() en la matriz de secuencias dada, con un tiempo de espera especificado por tv_sec y tv_usec
stream_set_blocking -- Establecer modo de bloqueo/no-bloqueo sobre una secuencia
stream_set_timeout -- Establecer el periodo de espera de una secuencia
stream_set_write_buffer -- Establece el uso de búferes de archivo en la secuencia dada
stream_socket_accept -- Aceptar una conexión en un socket creado por **stream_socket_server()**
stream_socket_client -- Abrir una conexión de socket de dominio de Internet o Unix
stream_socket_enable_crypto -- Turns encryption on/off on an already connected socket
stream_socket_get_name -- Recuperar el nombre de los sockets locales o remotos
stream_socket_pair -- Creates a pair of connected, indistinguishable socket streams
stream_socket_recvfrom -- Recibe datos desde un socket, conectado o no
stream_socket_sendto -- Envía un mensaje a un socket, sin importar si está conectado o no
stream_socket_server -- Crear un socket de servidor de dominio de Internet o Unix
stream_wrapper_register -- Registrar una envoltura URL implementada como una clase PHP
stream_wrapper_restore -- Restores a previously unregistered built-in wrapper
stream_wrapper_unregister -- Unregister a URL wrapper

stream_context_create

(PHP 4 >= 4.3.0, PHP 5)

stream_context_create -- Crear un contexto de secuencia

Descripción

resource **stream_context_create** ([array opciones])

Crea y devuelve un contexto de secuencia con cualquier número de opciones dadas en el conjunto *opciones*.

Las *opciones* deben venir en una matriz asociativa de matrices asociativas en el formato *$matriz ['envoltura']['opcion'] = $valor*. Su valor predeterminado es una matriz vacía.

Ejemplo 1. Uso de stream_context_create()

```
<?php
$opciones = array(
  'http'=>array(
    'method'=>"GET",
    'header'=>"Accept-language: en\r\n" .
              "Cookie: foo=bar\r\n"
  )
);

$contexto = stream_context_create($opciones);

/* Envia una peticion http a www.example.com con las cabeceras
 * adicionales mostradas anteriormente */
$da = fopen('http://www.example.com', 'r', false, $contexto);
fpassthru($da);
fclose($da);
?>
```

Vea también **stream_context_set_option()**, y el listado de envolturas soportadas con opciones de contexto (Apéndice L).

stream_context_get_default

(no version information, might be only in CVS)

stream_context_get_default -- Retreive the default streams context

Description

resource **stream_context_get_default** ([array options])

Returns the default stream context which is used whenever file operations (**fopen()**, **file_get_contents()**, etc...) are called without a context parameter. Options for the default context can optionally be specified with this function using the same syntax as **stream_context_create()**.

options must be an associative array of associative arrays in the format *$arr['wrapper']['option'] = $value*.

Ejemplo 1. Using stream_context_get_default()

```
<?php
$default_opts = array(
  'http'=>array(
    'method'=>"GET",
    'header'=>"Accept-language: en\r\n" .
              "Cookie: foo=bar",
    'proxy'=>"tcp://10.54.1.39:8000"
  )
);


$alternate_opts = array(
  'http'=>array(
    'method'=>"POST",
    'header'=>"Content-type: application/x-www-form-urlencoded\r\n" .
              "Content-length: " . strlen("baz=bomb"),
    'content'=>"baz=bomb"
  )
);

$default = stream_context_get_default($default_opts);
$alternate = stream_context_create($alternate_opts);

/* Sends a regular GET request to proxy server at 10.54.1.39
 * For www.example.com using context options specified in $default_opts
 */
readfile('http://www.example.com');

/* Sends a POST request directly to www.example.com
 * Using context options specified in $alternate_opts
 */
readfile('http://www.example.com', false, $alternate);

?>
```

See also **stream_context_create()**, and Listing of supported wrappers with context options (Apéndice L).

stream_context_get_options

(PHP 4 >= 4.3.0, PHP 5)

stream_context_get_options -- Recuperar las opciones para una secuencia/envoltura/contexto

Descripción

array **stream_context_get_options** (resource secuencia|contexto)

Devuelve una matriz de opciones de la secuencia o contexto especificado.

stream_context_set_option

(PHP 4 >= 4.3.0, PHP 5)

stream_context_set_option -- Establece una opción para una secuencia/envoltura/contexto

Descripción

bool **stream_context_set_option** (resource contexto|secuencia, string envoltura, string opcion, mixed valor)

Establece una opción en el contexto especificado. *valor* recibe el valor de *opcion* para la *envoltura*.

stream_context_set_params

(PHP 4 >= 4.3.0, PHP 5)

stream_context_set_params -- Establecer parámetros para una secuencia/envoltura/contexto

Descripción

bool **stream_context_set_params** (resource secuencia|contexto, array parametros)

parametros debe ser una matriz asociativa de la estructura: *$parametros['nombre_parametro'] = "valor_parametro";*.

Tabla 1. Parámetros

Parámetros	Propósito
notification	Nombre de la llamada de retorno definida por el usuario a ser llamada cuando una secuencia genere una notificación.

stream_copy_to_stream

(PHP 5)

stream_copy_to_stream -- Copia datos desde una secuencia a otra

Descripción

int **stream_copy_to_stream** (resource fuente, resource destino [, int longitud_maxima])

Crea una copia de hasta *longitud_maxima* bytes de datos a partir de la posición actual en *fuente* y la direcciona a *destino*. Si *longitud_maxima* no se especifica, será copiado todo el contenido restante en *fuente*. Devuelve el conteo total de bytes copiados.

Ejemplo 1. Ejemplo de stream_copy_to_stream()

```
<?php
$fuente = fopen('http://www.example.com', 'r');
$dest1  = fopen('primer1k.txt', 'w');
$dest2  = fopen('resto.txt', 'w');

echo stream_copy_to_stream($fuente, $dest1, 1024) . " bytes copiados a primer1k.txt\n";
echo stream_copy_to_stream($fuente, $dest2) . " bytes copiados a resto.txt\n";

?>
```

Vea también **copy()**.

stream_filter_append

(PHP 4 >= 4.3.0, PHP 5)

stream_filter_append -- Adjuntar un filtro a una secuencia

Descripción

bool **stream_filter_append** (resource secuencia, string nombre_filtro [, int lectura_escritura [, mixed parametros]])

Agrega el *nombre_filtro* a la lista de filtros adjuntos a la *secuencia*. Este filtro será añadido con los *parametros* especificados al *final* de la lista y por lo tanto será llamado al último durante las operaciones de la secuencia. Para agregar un filtro al comienzo de la lista, use **stream_filter_prepend()**.

Por defecto, **stream_filter_append()** adjuntará el filtro a la *cadena de filtros de lectura* si el archivo fue abierto para lectura (esto quiere decir, Modo del Archivo: *r*, o +). El filtro también se adjuntará a la *cadena de filtros de escritura* si el archivo fue abierto para escritura (esto quiere decir, Modo del Archivo: *w*, *a*, O +). Las constantes **STREAM_FILTER_READ**, **STREAM_FILTER_WRITE**, o **STREAM_FILTER_ALL** pueden ser pasadas también al parámetro *lectura_escritura* para sobrescribir este comportamiento.

Ejemplo 1. Control del lugar en el que se aplican los filtros

```
<?php
/* Abrir un archivo de prueba para lectura y escritura */
$da = fopen("prueba.txt", "rw");

/* Aplicar el filtro ROT13 a la cadena de filtros de escritura, pero
 * no a la cadena de filtros de lectura */
stream_filter_append($da, "string.rot13", STREAM_FILTER_WRITE);

/* Escribir una cadena simple al archivo, la cual sera transformada
   mediante ROT13 en su camino de salida */
fwrite($da, "Esta es una prueba\n");

/* Volver al inicio del archivo */
rewind($da);

/* Leer los contenidos del archivo de vuelta. Si hubiesemos aplicado
 * el filtro a la cadena de filtros de lectura tambien, veriamos el
 * texto de vuelta a su estado original debido a la retransformacion
 * con ROT133 */
fpassthru($da);

fclose($da);

/* Salida Esperada
   ---------------

Rfgn rf han cehron

 */
?>
```

Cuando se usan filtros personalizados (de usuario): La función **stream_filter_register()** debe ser llamada primero para registrar el filtro de usuario deseado para *nombre_filtro*.

Nota: Los datos de la secuencia son leídos desde los recursos (tanto locales como remotos) en paquetes, usando búferes internos para conservar todos los datos sin consumir. Cuando un nuevo filtro es agregado a la secuencia, los datos en los búferes internos son procesados a través del nuevo filtro en ese momento. Este comportamiento difiere de aquél de **stream_filter_prepend()**.

Vea también **stream_filter_register()**, y **stream_filter_prepend()**.

stream_filter_prepend

(PHP 4 >= 4.3.0, PHP 5)

stream_filter_prepend -- Adjuntar un filtro a una secuencia

Descripción

bool **stream_filter_prepend** (resource secuencia, string nombre_filtro [, int lectura_escritura [, mixed parametros]])

Agrega el *nombre_filtro* a la lista de filtros adjuntos a la *secuencia*. Este filtro será añadido con los *parametros* especificados al *comienzo* de la lista y por lo tanto será llamado al inicio de las operaciones de secuencia. Para añadir un filtro al final de la lista, use **stream_filter_append()**.

Por defecto, **stream_filter_prepend()** adjuntará el filtro a la *cadena de filtros de lectura* si el archivo fue abierto para lectura (esto quiere decir, Modo del Archivo: *r*, o +). El filtro será adjunto

también a la *cadena de filtros de escritura* si el archivo fue abierto para escritura (esto quiere decir, Modo del Archivo: *w*, *a*, o +). Las constantes **`STREAM_FILTER_READ`**, **`STREAM_FILTER_WRITE`**, o **`STREAM_FILTER_ALL`** también pueden ser pasadas al parámetro *lectura_escritura* para sobrescribir este comportamiente. Vea **stream_filter_append()** para consultar un ejemplo de éste parámetro.

> **Cuando se usan filtros personalizados (de usuario):** La función **stream_filter_register()** debe ser llamada primero para registrar el filtro de usuario deseado para *nombre_filtro*.
>
> **Nota:** Los datos de las secuencias son leídos desde los recursos (tanto locales como remotos) en paquetes, usando búferes internos para almacenar datos sin consumir. Cuando un nuevo filtro es agregado al inicio de la lista de filtros en una secuencia, los datos de los búferes internos, que ya han sido procesados con otros filtros, *no* serán reprocesados a través del nuevo filtro en ese momento. Este comportamiento difiere de aquél de **stream_filter_append()**.

Vea también **stream_filter_register()**, y **stream_filter_append()**.

stream_filter_register

(PHP 5)

stream_filter_register -- Registrar un filtro de secuencia implementado como una clase PHP derivada de *php_user_filter*

Descripción

bool **stream_filter_register** (string nombre_filtro, string nombre_clase)

stream_filter_register() le permite implementar su propio filtro en cualquier secuencia registrada utilizada con todas los otras funciones de sistema de archivos (tales como **fopen()**, **fread()** etc.).

Para implementar un filtro, necesita definir una clase como una extensión de *php_user_filter* con un número de funciones miembro, tal y como se define más adelante. Cuando realice operaciones de lectura/escritura en la secuencia a la que se ha adjuntado su filtro, PHP pasará los datos a través de su filtro (y cualquier otro filtre adjunto a esa secuencia) de modo que los datos puedan ser modificados como lo desee. Debe implementar los métodos exactamente como se describe más adelante - hacerlo de otra forma llevará a comportamientos indefenidos.

stream_filter_register() devolverá **`FALSE`** si *nombre_filtro* ya se encuentra definido.

int **filter** (resource entrada, resource salida, int &consumido, bool cerrando)

Este método es llamado siempre que se lean o escriban datos desde y hacia la secuencia adjunta (tal y como sucede con **fread()** o **fwrite()**). *entrada* es un recurso que apunta a una *brigada de paquetes* que contiene uno o más objetos de tipo *paquete* que contienen datos a ser filtrados. *salida* es un recurso que apunta a una segunda *brigada de paquetes* en la que deben ser colocados sus paquetes modificados. *consumido*, que debe declararse *siempre* por referencia, debe ser incrementado de acuerdo a la longitud de los datos que su filtro lee y altera. En la mayoría de casos esto quiere decir que usted incrementa *consumido* en $paquete->datalen para cada $paquete. Si la secuencia está en el proceso de ser cerrada (y por lo tanto este es el último paso por la cadena de filtros), el parámetro

cerrando será definido como **TRUE**. El método `filter` debe devolver uno de tres valores cuando complete su ejecución.

Valor de Retorno	Significado
PSFS_PASS_ON	El filtro fue procesado satisfactoriamente con los datos disponibles en la *brigada de paquetes salida*.
PSFS_FEED_ME	El filtro fue procesado satisfactoriamente, sin embargo, no habían datos disponibles para devolver. Se requieren más datos desde la secuencia o desde el filtro anterior.
PSFS_ERR_FATAL (predeterminado)	El filtro experimentó un error irrecuperable y no puede continuar.

void **onCreate** (void)

Este método es llamado durante la instanciación del objeto clase del filtro. Si su filtro ubica o inicializa cualquier otro tipo de recursos (como un búfer), éste es el lugar para hacerlo. Su implementación de este método debería devolver **FALSE** en caso de fallo, o **TRUE** si tiene éxito.

Cuando su filtro es instanciado por primera vez, y *su_filtro->onCreate()* es llamado, se colocarán a su disposición un número de propiedades, como lo muestra la siguiente tabla.

Propiedad	Contenidos
ClaseFiltro->filtername	Una cadena que contiene el nombre con el que fue instanciado el filtro. Los filtros pueden ser registrados bajo múltiples nombres o bajo comodines. Use ésta propiedad para determinar el nombre que fue usado.
ClaseFiltro->params	Los contenidos del parámetro *parametros* pasado a **stream_filter_append()** o **stream_filter_prepend()**.

void **onClose** (void)

Este método es llamado cuando se realiza la destrucción del filtro (por lo general, esto ocurre también durante la destrucción de la secuencia), y es ejecutado *después* de que el método *flush* es llamado. Si se ubicaron o inicializaron recursos durante *onCreate*, éste sería el momento para destruirlos o desecharlos.

El siguiente ejemplo implementa un filtro llamado *strtoupper* en la secuencia *foo-bar.txt*, el cual convierte a mayúsculas todos los caracteres alfabéticos escritos hacia/leidos desde esa secuencia.
Ejemplo 1. Filtro para convertir los caracteres a mayúsculas en la secuencia foo-bar.txt

```
<?php

/* Definicion de nuestra clase de filtro */
class strtoupper_filter extends php_user_filter {
  function filter($entrada, $salida, &$consumido, $cerrando)
  {
    while ($paquete = stream_bucket_make_writeable($entrada)) {
      $paquete->data = strtoupper($paquete->data);
      $consumido += $paquete->datalen;
      stream_bucket_append($salida, $paquete);
    }
    return PSFS_PASS_ON;
  }
}

/* Registrar nuestro filtro con PHP */
stream_filter_register("strtoupper", "strtoupper_filter")
    or die("Fall&oacute; el registro del filtro");

$da = fopen("foo-bar.txt", "w");

/* Adjuntar el filtro registrado a la secuencia que acabamos de abrir */
stream_filter_append($da, "strtoupper");

fwrite($da, "Linea1\n");
fwrite($da, "Palabra - 2\n");
fwrite($da, "Tan sencillo como 123\n");

fclose($da);

/* Leer los contenidos de vuelta
 */
readfile("foo-bar.txt");

?>
```

Salida

```
LINEA1
PALABRA - 2
TAN SENCILLO COMO 123
```

Ejemplo 2. Registro de una clase genérica de filtro para que coincida con múltiples nombres de filtro.

```php
<?php

/* Definicion de nuestra clase de filtro */
class filtro_cadena extends php_user_filter {
  var $modo;

  function filter($entrada, $salida, &$consumido, $cerrando)
  {
    while ($paquete = stream_bucket_make_writeable($entrada)) {
      if ($this->modo == 1) {
        $paquete->data = strtoupper($paquete->data);
      } elseif ($this->modo == 0) {
        $paquete->data = strtolower($paquete->data);
      }

      $consumido += $paquete->datalen;
      stream_bucket_append($salida, $paquete);
    }
    return PSFS_PASS_ON;
  }

  function onCreate()
  {
    if ($this->filtername == 'str.toupper') {
      $this->modo = 1;
    } elseif ($this->filtername == 'str.tolower') {
      $this->modo = 0;
    } else {
      /* Se ha pedido otro filtro str.* , reportar un fallo de modo
       * que PHP continue buscando */
      return false;
    }

    return true;
  }
}

/* Registrar nuestro filtro con PHP */
stream_filter_register("str.*", "filtro_cadena")
    or die("Fall&oacute; el registro del filtro");

$da = fopen("foo-bar.txt", "w");

/* Adjuntar el filtro registrado a la secuencia recien
 * abierta. Alternativamente, aqui podemos enlazar str.tolower */
stream_filter_append($da, "str.toupper");

fwrite($da, "Linea1\n");
fwrite($da, "Palabra - 2\n");
fwrite($da, "Tan sencillo como 123\n");

fclose($da);

/* Leer los contenidos de vuelta
 */
readfile("foo-bar.txt");

/* Salida
 * ------

LINEA1
PALABRA - 2
TAN SENCILLO COMO 123

 */

?>
```

Vea también **stream_wrapper_register()**, **stream_filter_prepend()**, y **stream_filter_append()**.

stream_filter_remove

(no version information, might be only in CVS)

stream_filter_remove -- Remove a filter from a stream

Description

bool **stream_filter_remove** (resource stream_filter)

Removes a stream filter previously added to a stream with **stream_filter_prepend()** or **stream_filter_append()**. Any data remaining in the filter's internal buffer will be flushed through to the next filter before removing it.

Ejemplo 1. Dynamicly refiltering a stream

```
<?php
/* Open a test file for reading and writing */
$fp = fopen("test.txt", "rw");

$rot13_filter = stream_filter_append($fp, "string.rot13", STREAM_FILTER_WRITE);
fwrite($fp, "This is ");
stream_filter_remove($rot13_filter);
fwrite($fp, "a test\n");

rewind($fp);
fpassthru($fp);
fclose($fp);

/* Expected Output
   ---------------

Guvf vf a test

 */
?>
```

See also **stream_filter_register()**, **stream_filter_append()**, and **stream_filter_prepend()**.

stream_get_contents

(PHP 5)

stream_get_contents -- Lee el resto de una secuencia en una cadena

Descripción

string **stream_get_contents** (resource gestor [, int longitud_maxima])

Función idéntica a **file_get_contents()**, excepto que **stream_get_contents()** opera en un recurso de archivo ya abierto y devuelve el contenido restante, hasta un máximo de *longitud_maxima* bytes, en una cadena.

> **Nota:** Esta función es segura binariamente.

Vea también **fgets()**, **fread()**, y **fpassthru()**.

stream_get_filters

(PHP 5)

stream_get_filters -- Recuperar la lista de filtros registrados

Descripción

array **stream_get_filters** (void)

Devuelve una matriz indexada que contiene el nombre de todos los filtros de secuencia disponibles en el sistema actual.

Ejemplo 1. Uso de stream_get_filters()

```
<?php
$lista_de_secuencias = stream_get_filters();
print_r($lista_de_secuencias);
?>
```

La salida será similar a la siguiente. Nota: pueden haber más o menos filtros en su versión de PHP.

```
Array (
  [0] => string.rot13
  [1] => string.toupper
  [2] => string.tolower
  [3] => string.base64
  [4] => string.quoted-printable
)
```

Vea también **stream_filter_register()**, y **stream_get_wrappers()**.

stream_get_line

(PHP 5)

stream_get_line -- Obtiene una línea desde un recurso de secuencia, hasta un delimitador dado

Descripción

string **stream_get_line** (resource gestor, int longitud, string final)

Devuelve una cadena de hasta *longitud* bytes leidos desde el archivo apuntado por *gestor*. La lectura finaliza cuando *longitud* bytes han sido leidos, cuando la cadena especificado por *final* es encontrada (la cual *no* es incluida en el valor de retorno), o al llegar al final del archivo, EOF, (lo que ocurra primero).

Si ocurre un error, devuelve **FALSE**.

Esta función es casi idéntica a **fgets()**, excepto que permite el uso de delimitadores de línea diferentes a los estándar \n, \r, y \r\n, y *no* devuelve el delimitador mismo.

Vea también **fread()**, **fgets()**, y **fgetc()**.

stream_get_meta_data

(PHP 4 >= 4.3.0, PHP 5)

stream_get_meta_data -- Recupera meta datos/cabeceras desde apuntadores a secuencias/archivos

Descripción

array **stream_get_meta_data** (resource secuencia)

Devuelve información sobre una *secuencia* existente. La secuencia puede ser cualquiera creada por **fopen()**, **fsockopen()** y **pfsockopen()**. La matriz resultante contiene los siguientes items:

- *timed_out* (bool) - **TRUE** si la secuencia ha superado el tiempo de espera máximo mientras esperaba datos en el último llamado a **fread()** o **fgets()**.
- *blocked* (bool) - **TRUE** si la secuencia se encuentra en modo de bloqueo de E/S. Vea **stream_set_blocking()**.
- *eof* (bool) - **TRUE** si la secuencia ha alcanzado el final del archivo. Note que para secuencias de socket, este miembro puede ser **TRUE** incluso cuando *unread_bytes* es diferente de cero. Para determinar si hay más datos para ser leidos, use **feof()** en lugar de leer este item.
- *unread_bytes* (int) - el número de bytes contenidos actualmente en el búfer de lectura.

Los siguientes items fueron agregados en PHP 4.3:

- *stream_type* (string) - una etiqueta que decribe la implementación de bajo nivel de la secuencia.
- *wrapper_type* (string) - una etiqueta que describe la implementación de envoltura de protocolo que cubre a la secuencia. Vea Apéndice L para más información sobre las envolturas.
- *wrapper_data* (mixed) - datos específicos de envoltura adjuntos a esta secuencia. Vea Apéndice L para más información sobre las envolturas y sus datos de envoltura.
- *filters* (array) - una matriz que contiene los nombres de cualquier filtro que esté apilado para esta secuencia. La documentación sobre los filtros puede encontrarse en el apéndice de Filtros.

  Nota: Esta función fue introducida en PHP 4.3, pero anteriormente, **socket_get_status()** podía ser usado para recuperar los primeros cuatro items, para *secuencias basadas en sockets únicamente*.

  En PHP 4.3 y versiones posteriores, **socket_get_status()** es un alias de esta función.

  Nota: Esta función NO trabaja sobre sockets creados por la Extensión de sockets.

Los siguientes ítems fueron agregados en PHP 5.0:

- *mode* (string) - el modo (o permisos) de la URI asociada con ésta secuencia.

- *seakable* (bool) - si la secuencia actual puede ser reubicada.
- *uri* (string) - el nombre de archivo/URI asociado con ésta secuencia.

stream_get_transports

(PHP 5)

stream_get_transports -- Recuperar la lista de transportes de socket registrados

Descripción

array **stream_get_transports** (void)

Devuelve una matriz indexada que contiene los nombres de todos los transportes de socket disponibles en el sistema actual.

Ejemplo 1. Uso de stream_get_transports()

```
<?php
$lista = stream_get_transports();
print_r($lista);
?>
```

La salida será similar a la siguiente. Nota: pueden haber más o menos transportes en su versión de PHP.

```
Array (
  [0] => tcp
  [1] => udp
  [2] => unix
  [3] => udg
)
```

Vea también **stream_get_filters()**, y **stream_get_wrappers()**.

stream_get_wrappers

(PHP 5)

stream_get_wrappers -- Recuperar la lista de secuencias registradas

Descripción

array **stream_get_wrappers** (void)

Devuelve una matriz indexada que contiene los nombres de todas las envolturas de secuencia disponibles en el sistema actual.

Vea también **stream_wrapper_register()**.

stream_register_wrapper

stream_register_wrapper -- Alias de **stream_wrapper_register()**

Descripción

Esta función es un alias de **stream_wrapper_register()**. Esta función es incluida por razones de compatibilidad con PHP 4.3.0 y PHP 4.3.1 únicamente. **stream_wrapper_register()** debería ser usada en su lugar.

stream_select

(PHP 4 >= 4.3.0, PHP 5)

stream_select -- Ejecuta el ñalente al llamado de sistema select() en la matriz de secuencias dada, con un tiempo de espera especificado por tv_sec y tv_usec

Descripción

int **stream_select** (array &lectura, array &escritura, array &excepcional, int tv_sec [, int tv_usec])

La función **stream_select()** acepta una matriz de secuencias y espera a que éstas cambien su status. Su operación es ñalente a la de la función **socket_select()**, excepto que actúa sobre secuencias.

Las secuencias listadas en la matriz *lectura* serán vigiladas para ver si aparecen caracteres disponibles para lectura (o más precisamente, para ver si una operación de lectura no producirá un bloqueo - en particular, un recurso de secuencia se encuentra listo también al llegar al final del archivo, en cuyo caso un llamado a **fread()** devolverá una cadena de longitud cero).

Las secuencias listadas en la matriz *escritura* serán vigiladas para ver si una escritura no crea bloqueos.

Las secuencias listadas en la matriz *excepcional* serán vigiladas por la llegada de datos excepcionales ("out-of-band") de alta prioridad.

> **Nota:** Cuando **stream_select()** devuelve un valor, las matrices *lectura*, *escritura* y *excepcional* son modificadas para indicar cuáles recursos de secuencia modificaron su status en realidad.

Los parámetros *tv_sec* y *tv_usec*, juntos forman el parámetro *tiempo de espera*, *tv_sec* especifica el número de segundos, mientras que *tv_usec* el número de microsegundos. El *tiempo de espera* es un límite superior sobre la cantidad de tiempo que **stream_select()** esperará antes de devolver un valor. Si tanto *tv_sec* como *tv_usec* son definidos como *0*, **stream_select()** no esperará por datos - en su lugar devolverá un valor inmediatamente, indicando el status actual de las secuencias. Si *tv_sec* es **`NULL`** **stream_select()** puede crear un bloqueo indefinidamente, y sólo devolverá un valor cuando ocurra un evento en alguna de las secuencias vigiladas (o si una señal interrumpe el llamado de sistema).

En caso de éxito, **stream_select()** devuelve el número de recursos de secuencia modificados contenidos en las matrices, que puede ser cero si el tiempo de espera expira antes de que algo interesante suceda. En caso de fallo, el valor **`FALSE`** es devuelto y se genera una advertencia (esto puede pasar si el llamado de sistema es interrumpido por una señal entrante).

Aviso

El uso de un valor de tiempo de espera de *0* le permite consultar el status de las secuencias de forma instantánea, sin embargo, NO es buena idea usar un valor de tiempo de espera de *0* en un ciclo, dado que causará que su script consuma mucho tiempo de CPU.

Es mucho mejor especificar un valor de tiempo de espera de algunos pocos segundos, aunque si necesita hacer chequeos y ejecutar otro segmento de código concurrentemente, usar un valor de tiempo de espera de por lo menos *200000* microsegundos le ayudará a reducir el uso de CPU de su script.

Recuerde que el valor de tiempo de espera es el tiempo máximo que transcurrirá; **stream_select()** devolverá un valor tan pronto como las secuencias solicitadas se encuentren listas para su uso.

No necesita pasar cada matriz a **stream_select()**. Puede dejar de especificarlas y usar una matriz vacía o **NULL** en su lugar. Tampoco olvide que éstas matrices son pasadas *por referencia* y serán modificadas después de que **stream_select()** devuelva un valor.

Este ejemplo revisa si han llegado datos para lectura ya sea en *$secuencia1* o en *$secuencia2*. Dado que el valor de tiempo de espera es *0*, la función retornará inmediatamente:

```
<?php
/* Preparar la matriz de lectura */
$lectura = array($secuencia1, $secuencia2);

if (false === ($num_secuencias_modificadas = stream_select($lectura, $escritura = NULL,
    /* Gestion de errores */
} elseif ($num_secuencias_modificadas > 0) {
    /* En por lo menos una de las secuencias ha ocurrido algo interesante */
}
?>
```

Nota: Debido a una limitación en el Motor Zend actual, no es posible pasar un modificador constante como **NULL** directamente como parámetro a una función que espera que tal parámetro sea pasado por referencia. En su lugar, use una variable temporal o una expresión en donde el miembro del lado izquierdo sea una variable temporal:

```
<?php
stream_select($l, $e, $ex = NULL, 0);
?>
```

Nota: Asegúrese de usar el operador === cuando efectúe chequeos de error. Dado que **stream_select()** puede devolver 0, la comparación con == evaluaría a **TRUE**:

```
<?php
if (false === stream_select($l, $e, $ex = NULL, 0)) {
    echo "stream_select() fall&oacute;\n";
}
?>
```

Nota: Si lee/escribe sobre una secuencia devuelta con las matrices, tenga en cuenta que éstas no necesariamente leen/escriben la cantidad completa de datos que usted ha solicitado. Prepárese para gestionar incluso la lectura/escritura de un solo byte.

Compatibilidad con Windows: El uso de **stream_select()** sobre un pipe devuelto desde **proc_open()** puede causar pérdida de datos bajo Windows 98.

El uso de **stream_select()** con descriptores de archivo devueltos por **proc_open()** fallará y devolverá **FALSE** bajo Windows.

Vea también **stream_set_blocking()**.

stream_set_blocking

(PHP 4 >= 4.3.0, PHP 5)

stream_set_blocking -- Establecer modo de bloqueo/no-bloqueo sobre una secuencia

Descripción

bool **stream_set_blocking** (resource secuencia, int modo)

Si *modo* es **FALSE**, la secuencia dada será colocada en modo de no-bloqueo, y si es **TRUE**, será colocada en modo de bloqueo. Esto afecta llamadas como **fgets()** y **fread()** que leen datos desde la secuencia. En modo de no-bloqueo, una llamada a **fgets()** siempre retornará inmediatamante, mientras que en modo de bloqueo esperará a que hayan datos disponibles en la secuencia.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Esta función era llamada anteriormente **set_socket_blocking()**, y más adelante **socket_set_blocking()** pero su uso bajo estos nombres se considera obsoleto.

> **Nota:** Antes de PHP 4.3, ésta función solo trabajaba sobre secuencias basadas en sockets. A partir de PHP 4.3, esta función trabaja sobre cualquier secuencia que soporte el modo de no-bloqueo (en la actualidad, archivos regulares y secuencias de socket).

Vea también **stream_select()**.

stream_set_timeout

(PHP 4 >= 4.3.0, PHP 5)

stream_set_timeout -- Establecer el periodo de espera de una secuencia

Descripción

bool **stream_set_timeout** (resource secuencia, int segundos [, int microsegundos])

Establece el valor de tiempo de espera sobre la *secuencia*, expresada como la suma de *segundos* y *microsegundos*. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplo 1. Ejemplo de stream_set_timeout()

```
<?php
$da = fsockopen("www.example.com", 80);
if (!$da) {
    echo "No fue posible abrir\n";
} else {
    fwrite($da, "GET / HTTP/1.0\n\n");
    stream_set_timeout($da, 2);
    $res = fread($da, 2000);
    var_dump(stream_get_meta_data($da));
    fclose($da);
    echo $res;
}
?>
```

> **Nota:** A partir de PHP 4.3, esta función puede (potencialmente) trabajar sobre cualquier

clase de secuencia. En PHP 4.3, las secuencias basadas en sockets son aun el único tipo soportado por el núcleo de PHP, aunque las secuencias de otras extensiones pueden soportar esta función.

Esta función era llamada anteriormente **set_socket_timeout()** y más adelante **socket_set_timeout()**, pero su uso bajo estos nombres se considera obsoleto.

Vea también **fsockopen()** y **fopen()**.

stream_set_write_buffer

(PHP 4 >= 4.3.0, PHP 5)

stream_set_write_buffer -- Establece el uso de búferes de archivo en la secuencia dada

Descripción

int **stream_set_write_buffer** (resource secuencia, int bufer)

La salida al usar **fwrite()** se acumula normalmente en un búfer a los 8K. Esto quiere decir que si existen dos procesos que desean escribir sobre la misma secuencia de salida (un archivo), cada uno es pausado después de los 8K de datos para permitir que el otro escriba. **stream_set_write_buffer()** establece el uso del búfer para operaciones de escritura sobre el apuntador de archivo dado *secuencia* a *bufer* bytes. Si *bufer* es 0, entonces las operaciones de escritura no usan el búfer. Esto se asegura de que todas las escrituras con **fwrite()** sean completadas antes de que a otros procesos les sea permitido escribir sobre esa secuencia de salida.

La función devuelve 0 en caso de éxito, o EOF si la petición no puede ser cumplida.

El siguiente ejemplo demuestra el modo de usar **stream_set_write_buffer()** para crear una secuencia sin uso de búferes.

Ejemplo 1. Ejemplo de stream_set_write_buffer()

```
<?php
$da = fopen($archivo, "w");
if ($da) {
  stream_set_write_buffer($da, 0);
  fwrite($da, $salida);
  fclose($da);
}
?>
```

Vea también **fopen()** y **fwrite()**.

stream_socket_accept

(PHP 5)

stream_socket_accept -- Aceptar una conexión en un socket creado por **stream_socket_server()**

Descripción

resource **stream_socket_accept** (resource socket_servidor [, float tiempo_espera [, string &nombre_de_par]])

Aceptar una conexión en un socket creado previamente por **stream_socket_server()**. Si se especifica *tiempo_espera*, el tiempo de espera predeterminado para el proceso accept del socket será sobrescrito con el tiempo especificado en segundos. El nombre (dirección) del cliente que se ha conectado será pasado de vuelta en *nombre_de_par*, si se incluye y se encuentra disponible en el transporte seleccionado.

Puede determinarse el valor de *nombre_de_par* también más adelante, usando **stream_socket_get_name()**.

Si el llamado falla, devolverá **FALSE**.

Aviso
El uso de esta función con sockets de servidor tipo UDP es un error. Deben usarse **stream_socket_recvfrom()** y **stream_socket_sendto()** en su lugar.

Vea también **stream_socket_server()**, **stream_socket_get_name()**, **stream_set_blocking()**, **stream_set_timeout()**, **fgets()**, **fgetss()**, **fwrite()**, **fclose()**, **feof()**, y la extensión Curl.

stream_socket_client

(PHP 5)

stream_socket_client -- Abrir una conexión de socket de dominio de Internet o Unix

Descripción

resource **stream_socket_client** (string socket_remoto [, int &errno [, string &errstr [, float tiempo_espera [, int banderas [, resource contexto]]]]])

Inicia una conexión secuenciada o tipo datagrama al destino especificado por *socket_remoto*. El tipo de socket creado es determinado por el transporte especificado usando el formato de URL estándar: *transporte://destino*. Para sockets de dominio de Internet (AF_INET) tales como TCP y UDP, la porción *destino* del parámetro *socket_remoto* debe consistir de un nombre de host o dirección IP seguido de un signo de dos puntos y un número de puerto. Para sockets de dominio Unix, la porción *destino* debe apuntar al archivo de socket en el sistema de archivos. El *tiempo_espera* opcional puede ser usado para establecer un tiempo de espera máximo en segundos para el llamado de conexión del sistema. *banderas* es un campo de máscara de bits que puede ser definido como una combinación de banderas de conexión. Actualmente, las banderas de conexión disponibles están limitadas a **STREAM_CLIENT_ASYNC_CONNECT** y **STREAM_CLIENT_PERSISTENT**.

> **Nota:** Si necesita establecer un tiempo de espera para la lectura/escritura de datos sobre el socket, use **stream_set_timeout()**, ya que el parámetro *tiempo_espera* de **stream_socket_client()** sólo se aplica cuando se establece la conexión con el socket.

stream_socket_client() devuelve un recurso de secuencia que puede ser usado junto con las otras funciones de archivos (tales como **fgets()**, **fgetss()**, **fwrite()**, **fclose()**, y **feof()**).

Si la llamada falla, devolverá **FALSE** y si los argumentos opcionales *errno* y *errstr* están presentes, éstos serán definidos de modo tal que indiquen el error de nivel de sistema actual que ocurrió en el llamado de nivel de sistema *connect()*. Si el valor devuelto en *errno* es *0* y la función devolvió **FALSE**, es un indicio de que el error ocurrió antes del llamado a *connect()*. Esto es por lo general debido a un problema con la inicialización del socket. Note que los argumentos *errno* y *errstr* serán siempre pasados por referencia.

Dependiendo del entorno, el dominio Unix o el tiempo de espera de conexión opcional pueden no estar disponibles. Una lista de transportes disponibles puede ser recuperada usando **stream_get_transports()**. Vea Apéndice N para consultar una lista de transportes incorporados.

La secuencia será abierta por defecto en modo de bloqueo. Puede modificarla a modo de no-bloqueo usando **stream_set_blocking()**.

Ejemplo 1. Ejemplo de stream_socket_client()

```
<?php
$da = stream_socket_client("tcp://www.example.com:80", $errno, $errstr, 30);
if (!$da) {
    echo "$errstr ($errno)<br />\n";
} else {
    fwrite($da, "GET / HTTP/1.0\r\nHost: www.example.com\r\nAccept: */*\r\n\r\n");
    while (!feof($da)) {
        echo fgets($da, 1024);
    }
    fclose($da);
}
?>
```

El siguiente ejemplo muestra cómo recuperar la fecha y hora desde el servicio UDP "daytime" (puerto 13) en su máquina local.

Ejemplo 2. Uso de una conexión UDP

```
<?php
$da = stream_socket_client("udp://127.0.0.1:13", $errno, $errstr);
if (!$da) {
    echo "ERROR: $errno - $errstr<br />\n";
} else {
    fwrite($da, "\n");
    echo fread($da, 26);
    fclose($da);
}
?>
```

Aviso
Los sockets UDP parecerán haber sido abiertos sin error en ocasiones, incluso si el host remoto no ha podido ser contactado. El error solo se hará aparente cuando lea o escriba datos desde/hacia el socket. La razón de esto es que UDP es un protocolo "sin conexión", lo que quiere decir que el sistema operativo no intenta establecer un enlace para el socket hasta que realmente necesita enviar o recibir datos.

> **Nota:** Cuando se especifique una dirección numérica IPv6 (p.ej. fe80::1) se debe incluir la IP entre corchetes. Por ejemplo *tcp://[fe80::1]:80*.

Vea también **stream_socket_server()**, **stream_set_blocking()**, **stream_set_timeout()**, **stream_select()**, **fgets()**, **fgetss()**, **fwrite()**, **fclose()**, **feof()**, y la extensión Curl.

stream_socket_enable_crypto

(no version information, might be only in CVS)

stream_socket_enable_crypto -- Turns encryption on/off on an already connected socket

Description

mixed **stream_socket_enable_crypto** (resource stream, bool enable [, int crypto_type [, resource session_stream]])

When called with the *crypto_type* parameter, **stream_socket_enable_crypto()** will setup encryption

on the stream using the specified method.

Valid values for *crypto_type*

- **STREAM_CRYPTO_METHOD_SSLv2_CLIENT**
- **STREAM_CRYPTO_METHOD_SSLv3_CLIENT**
- **STREAM_CRYPTO_METHOD_SSLv23_CLIENT**
- **STREAM_CRYPTO_METHOD_TLS_CLIENT**
- **STREAM_CRYPTO_METHOD_SSLv2_SERVER**
- **STREAM_CRYPTO_METHOD_SSLv3_SERVER**
- **STREAM_CRYPTO_METHOD_SSLv23_SERVER**
- **STREAM_CRYPTO_METHOD_TLS_SERVER**

Once the crypto settings are established, cryptography can be turned on and off dynamically by passing **TRUE** or **FALSE** in the *enable* parameter.

If this stream should be seeded with settings from an already established crypto enabled stream, pass that stream's resource variable in the fourth parameter.

Returns **TRUE** on success, **FALSE** if negotiation has failed or *0* if there isn't enough data and you should try again (only for non-blocking sockets).

Ejemplo 1. stream_socket_enable_crypto() Example

```
<?php
$fp = stream_socket_client("tcp://myproto.example.com:31337", $errno, $errstr, 30);
if (!$fp) {
  die("Unable to connect: $errstr ($errno)");
}
/* Turn on encryption for login phase */
stream_socket_enable_crypto($fp, true, STREAM_CRYPTO_METHOD_SSLv23_CLIENT);
fwrite($fp, "USER god\r\n");
fwrite($fp, "PASS secret\r\n");
/* Turn off encryption for the rest */
stream_socket_enable_crypto($fp, false);
while ($motd = fgets($fp)) {
  echo $motd;
}
fclose($fp);
?>
```

Referencia LXXXIX, *OpenSSL Functions*, and Apéndice N

stream_socket_get_name

(PHP 5)

stream_socket_get_name -- Recuperar el nombre de los sockets locales o remotos

Descripción

string **stream_socket_get_name** (resource gestor, bool desea_nombre_del_par)

Devuelve el nombre local o remoto de una conexión de socket dada. Si *desea_nombre_del_par* es definido como **TRUE**, el nombre de socket remoto será devuelto, si es definido como **FALSE**, se devolverá el nombre de socket local.

Vea también **stream_socket_accept()**.

stream_socket_pair

(no version information, might be only in CVS)

stream_socket_pair -- Creates a pair of connected, indistinguishable socket streams

Description

array **stream_socket_pair** (int domain, int type, int protocol)

stream_socket_pair() creates a pair of connected, indistinguishable socket streams. This function is commonly used in IPC (InterProcess Communication).

stream_socket_recvfrom

(PHP 5)

stream_socket_recvfrom -- Recibe datos desde un socket, conectado o no

Descripción

string **stream_socket_recvfrom** (resource socket, int longitud [, int banderas [, string &direccion]])

La función **stream_socket_recvfrom()** acepta datos desde un socket remoto, hasta una cantidad de *longitud* bytes. Si se provee una *direccion*, ésta será definida con la dirección del socket remoto.

El valor de *banderas* puede ser cualquier combinación de los siguientes:

Tabla 1. valores posibles para *banderas*

STREAM_OOB	Procesa datos OOB (out-of-band).
STREAM_PEEK	Recuperar datos desde el socket, pero no consumir el búfer. Llamadas subsiguientes a **fread()** o **stream_socket_recvfrom()** verán los mismos datos.

Ejemplo 1. Ejemplo de stream_socket_recvfrom()

```
<?php
/* Abrir un socket de servidor en el puerto 1234 en localhost */
$servidor = stream_socket_server('tcp://127.0.0.1:1234');

/* Aceptar una conexion */
$socket = stream_socket_accept($servidor);

/* Tomar un paquete (1500 es un tamanyo tipico) de datos OOB */
echo "Recibidos Out-Of-Band: '" . stream_socket_recvfrom($socket, 1500, STREAM_OOB) . "

/* Echar un vistazo a los datos en banda normales, pero no consumirlos. */
echo "Datos: '" . stream_socket_recvfrom($socket, 1500, STREAM_PEEK) . "'\n";

/* Recibir exactamente el mismo paquete de nuevo, pero eliminarlo del
 * bufer esta vez. */
echo "Datos: '" . stream_socket_recvfrom($socket, 1500) . "'\n";

/* Cerrarlo */
fclose($socket);
fclose($servidor);
?>
```

> **Nota:** Si un mensaje recibido tiene una longitud mayor que el parámetro *longitud*, los bytes sobrantes pueden ser descartados dependiendo del tipo de socket del mensaje recibido (como UDP).

Vea también **stream_socket_sendto()**, **stream_socket_client()**, y **stream_socket_server()**.

stream_socket_sendto

(PHP 5)

stream_socket_sendto -- Envía un mensaje a un socket, sin importar si está conectado o no

Descripción

int **stream_socket_sendto** (resource socket, string datos [, int banderas [, string direccion]])

La función **stream_socket_sendto()** envía los datos especificados por *datos* a través del socket especificado por *socket*. Será usada la dirección especificada cuando la secuencia de socket fue creada, a menos que una dirección alternativa sea especificada en *direccion*.

El valor de *banderas* puede ser cualquier combinación de los siguientes:

Tabla 1. valores posibles para *banderas*

STREAM_OOB	Procesar datos OOB (out-of-band).

Ejemplo 1. Ejemplo de stream_socket_sendto()

```
<?php
/* Abrir un socket en el puerto 1234 en localhost */
$socket = stream_socket_client('tcp://127.0.0.1:1234');

/* Enviar datos ordinarios mediante los canales ordinarios. */
fwrite($socket, "Transmision normal de datos.");

/* Enviar mas datos fuera de banda. */
stream_socket_sendto($socket, "Datos Out of Band.", STREAM_OOB);

/* Cerrarlo */
fclose($socket);
?>
```

Vea también **stream_socket_recvfrom()**, **stream_socket_client()**, y **stream_socket_server()**.

stream_socket_server

(PHP 5)

stream_socket_server -- Crear un socket de servidor de dominio de Internet o Unix

Descripción

resource **stream_socket_server** (string socket_local [, int &errno [, string &errstr [, int banderas [, resource contexto]]]])

Crea un socket secuenciado o tipo datagrama en el *socket_local* especificado. El tipo de socket creado es determinado por el transporte especificado usando el formato de URL estándar: *transporte://destino*. Para sockets de Dominio de Internet (AF_INET) tales como TCP y UDP, la porción *destino* del parámetro *socket_remoto* debe consistir de un nombre de host o dirección IP, seguido de un signo de dos puntos y un número de puerto. Para sockets de dominio Unix, la porción *destino* debe apuntar al archivo de socket en el sistema de archivos. *banderas* es un campo de máscara de bits que puede definirse como una combinación de banderas de creación de sockets. El valor predeterminado de banderas es **STREAM_SERVER_BIND** | **STREAM_SERVER_LISTEN**.

> **Nota:** Para sockets UDP, debe usar **STREAM_SERVER_BIND** como el parámetro *banderas*.

Esta función solo crea un socket, para empezar a aceptar conexiones use **stream_socket_accept()**.

Si la llamada falla, devolverá **FALSE** y si los argumentos opcionales *errno* y *errstr* están presentes, éstos serán definidos de forma tal que indiquen el error de nivel de sistema actual que ha ocurrido en los llamados de sistema *socket()*, *bind()*, y *listen()*. Si el valor devuelto en *errno* es *0* y la función ha devuelto **FALSE**, es un indicio de que el error ocurrió antes del llamado a *bind()*. Esto, por lo general, es debido a un problema en la inicialización del socket. Note que los argumentos *errno* y *errstr* siempre serán pasados por referencia.

Dependiendo del entorno, los sockets de dominio Unix pueden no estar disponibles. Una lista de transportes disponibles puede ser recuperada usando **stream_get_transports()**. Vea Apéndice N para consultar una lista de transportes integrados.

Ejemplo 1. Uso de sockets de servidor TCP

```
<?php
$socket = stream_socket_server("tcp://0.0.0.0:8000", $errno, $errstr);
if (!$socket) {
  echo "$errstr ($errno)<br />\n";
} else {
  while ($con = stream_socket_accept($socket)) {
    fwrite($con, 'La hora local es ' . date('n/j/Y g:i a') . "\n");
    fclose($con);
  }
  fclose($socket);
}
?>
```

El siguiente ejemplo muestra como actuar como servidor de hora, el cual puede responder a consultas sobre la hora actual, tal y como se muestra en un ejemplo ubicado en la página sobre **stream_socket_client()**.

> **Nota:** La mayoría de sistemas requieren acceso de root para crear un socket de servidor en un puerto menor a 1024.

Ejemplo 2. Uso de sockets de servidor UDP

```
<?php
$socket = stream_socket_server("udp://127.0.0.1:1113", $errno, $errstr, STREAM_SERVER_B
if (!$socket) {
    die("$errstr ($errno)");
}

do {
    $pqt = stream_socket_recvfrom($socket, 1, 0, $peer);
    echo "$peer\n";
    stream_socket_sendto($socket, date("D M j H:i:s Y\r\n"), 0,
    $peer);
} while ($pqt !== false);

?>
```

> **Nota:** Cuando se especifique una dirección numérica IPv6 (p.ej. fe80::1) se debe incluir la IP entre corchetes. Por ejemplo *tcp://[fe80::1]:80*.

Vea también **stream_socket_client()**, **stream_set_blocking()**, **stream_set_timeout()**, **fgets()**, **fgetss()**, **fwrite()**, **fclose()**, **feof()**, y la extensión Curl.

stream_wrapper_register

(PHP 4 >= 4.3.2, PHP 5)

stream_wrapper_register -- Registrar una envoltura URL implementada como una clase PHP

Descripción

bool **stream_wrapper_register** (string protocolo, string nombre_clase)

stream_wrapper_register() le permite implementar sus propios gestores y secuencias de protocolo para su uso con todas las funciones de sistema de archivos (tales como **fopen()**, **fread()** etc.).

Para implementar una envoltura, necesita definir una clase con un número de funciones miembro, tal y como se define más adelante. Cuando alguien abre su secuencia mediante fopen, PHP creará una instancia de *nombre_clase* y luego llamará algunos métodos en esa instancia. Debe implementar los métodos exactamente como se describe a continuación - de no ser así, producirá

comportamientos indefinidos.

> **Nota:** A partir de PHP 5.0.0, la instancia de *nombre_clase* será poblada con una propiedad *contexto* que hace referencia a un *Recurso de Contexto*, el cual puede obtenerse con **stream_context_get_options()**. Si no se ha pasado ningún contexto a la función de creación de secuencia, *contexto* será definido como **NULL**.

stream_wrapper_register() devolverá **FALSE** si el *protocolo* ya tiene un gestor.

bool **stream_open** (string ruta, string modo, int opciones, string ruta_abierta)

Este método es llamado inmediatamente después de que su objeto de secuencia es creado. *ruta* especifica la URL pasada a **fopen()** y que éste objeto supuestamente debe recuperar. Puede usar **parse_url()** para separar la ruta.

modo es el modo usado para abrir el archivo, tal y como se proporciona en **fopen()**. Usted es responsable por el chequeo de la validez del *modo* para la *ruta* solicitada.

opciones contiene banderas adicionales definidas por la interfaz de programación de las secuencias. Puede contener uno o más de los siguientes valores, unidos mediante la operación lógica OR.

Bandera	Descripción
STREAM_USE_PATH	Si *ruta* es relativa, buscar por el recurso usando include_path.
STREAM_REPORT_ERRORS	Si esta bandera está activa, usted se hace responsable por la generación de errores usando **trigger_error()** durante la apertura de la secuencia. Si esta bandera no está definida, no debe generar ningún error.

Si la *ruta* es abierta satisfactoriamente, y STREAM_USE_PATH es definida en *opciones*, usted debe definir *ruta_abierta* como la ruta completa hacia el archivo/recurso que fue abierto en realidad.

Si el recurso solicitado fue abierto satisfactoriamente, debe devolver **TRUE**, o **FALSE** de lo contrario.

void **stream_close** (void)

Este método es llamado cuando la secuencia es cerrada, usando **fclose()**. Debe liberar cualquier recurso que haya sido bloqueado o reservado por la secuencia.

string **stream_read** (int conteo)

Este método es llamado en respuesta a llamadas de **fread()** y **fgets()** en la secuencia. Usted debe devolver hasta *conteo* bytes de datos desde su posición actual de lectura/escritura como una cadena. Si hay menos de *conteo* bytes disponibles, devuelva tantos como pueda. Si no hay más datos disponibles, devuelva **FALSE** o una cadena vacía. También debe actualizar la posición de lectura/escritura de la secuencia en el número de bytes que fueron leídos con éxito.

int **stream_write** (string datos)

Este método es llamado en respuesta a llamadas de **fwrite()** en el sistema. Debe guardar *datos* en el modelo de almacenamiento base usado por su secuencia. Si no hay suficiente espacio disponible, intente almacenar tantos bytes como le sea posible. Debe devolver el número de bytes que fueron almacenados satisfactoriamente en la secuencia, o 0 si no fue posible almacenar ninguno. También

debe actualizar la posición de lectura/escritura de la secuencia en el número de bytes que fueron escritos con éxito.

bool **stream_eof** (void)

Este método es llamado en respuesta a llamados de **feof()** en la secuencia. Debe devolver **TRUE** si la posición de lectura/escritura se encuentra al final de la secuencia y no hay más datos disponibles para su lectura, o **FALSE** de lo contrario.

int **stream_tell** (void)

Este método es llamado en respuesta a llamados de **ftell()** en la secuencia. Debe devolver la posición actual de lectura/escritura en la secuencia.

bool **stream_seek** (int desplazamiento, int punto_partida)

Este método es llamado en respuesta a llamadas de **fseek()** en la secuencia. Debe actualizar la posición de lectura/escritua en la secuencia de acuerdo a *desplazamiento* y *punto_partida*. Vea **fseek()** para más información sobre éstos parámetros. Devuelva **TRUE** si la posición fue actualizada, **FALSE** de lo contrario.

bool **stream_flush** (void)

Este método es llamado en respuesta a llamadas de **fflush()** en la secuencia. Si ha usado un caché con los datos de su secuencia, pero no los ha guardado aun en el modelo de almacenamiento base, debe hacerlo ahora. Devuelva **TRUE** si los datos en caché fueron almacenados satisfactoriamente (o si no habían datos a almacenar), o **FALSE** si los datos no pudieron ser almacenados.

array **stream_stat** (void)

Este método es llamado en respuesta a llamadas de **fstat()** en la secuencia y debe devolver una matriz que contenga los mismos valores que sean apropiados para la secuencia.

bool **unlink** (string ruta)

Este método es llamado en respuesta a llamadas de **unlink()** sobre rutas URL asociadas con la envoltura y debe intentar la eliminación del item especificado por *ruta*. Debe devolver **TRUE** de tener éxito o **FALSE** en caso de fallo. Para asegurarse de que el mensaje de error correcto sea devuelto, no defina éste método si su envoltura no soporta eliminaciones.

> **Nota:** El método de envoltura en espacio de usuario unlink no es soportado antes de PHP 5.0.0.

bool **rename** (string ruta_fuente, string ruta_destino)

Este método es llamado en respuesta a llamadas de **rename()** en rutas URL asociadas con la envoltura y debe intentar renombrar el item especificado por *ruta_fuente* a la cadena dada por *ruta_destino*. Debe devolver **TRUE** en caso de éxito o **FALSE** en caso de fallo. Para asegurarse de que el mensaje de error correcto sea devuelto, no defina éste método si su envoltura no soporta el renombramiento de rutas.

> **Nota:** El método de envoltura en espacio de usuario rename no es soportado antes de PHP 5.0.0.

bool **mkdir** (string ruta, int modo, int opciones)

Este método es llamado en respuesta a llamadas de **mkdir()** en rutas URL asociadas con la envoltura y debe intentar crear el directorio especificado por *ruta*. Debe devolver **TRUE** de tener éxito o **FALSE** en caso de fallo. Para asegurarse de que el mensaje de error correcto sea devuelto, no defina éste método si su envoltura no soporta la creación de directorios. Valores posibles para *opciones* incluyen **STREAM_REPORT_ERRORS** y **STREAM_MKDIR_RECURSIVE**.

> **Nota:** El método de envoltura en espacio de usuario mkdir no es soportado antes de PHP 5.0.0.

bool **rmdir** (string ruta, int opciones)

Este método es llamado en respuesta a llamadas de **rmdir()** sobre rutas URL asociadas con la envoltura y debería intentar eliminar el directorio especificado por *ruta*. Debe devolver **TRUE** de tener éxito o **FALSE** en caso de fallo. Para asegurarse de que el mensaje de error correcto sea devuelto, no defina éste método si su envoltura no soporta la eliminación de directorios. Posibles valores para *opciones* incluyen **STREAM_REPORT_ERRORS**.

> **Nota:** El método de envoltura en espacio de usuario rmdir no es soportado antes de PHP 5.0.0.

bool **dir_opendir** (string ruta, int opciones)

Este método es llamado inmediatamente cuando su objeto de secuencia es creado para examinar contenidos de directorio con **opendir()**. *ruta* especifica la URL que fue pasada a **opendir()** y que éste objeto supuestamente va a explorar. Puede usar **parse_url()** para separar ésta ruta.

array **url_stat** (string ruta, int banderas)

Este método es llamado en respuesta a llamadas de **stat()** en rutas URL asociadas con la envoltura y debe devolver tantos elementos en común con la función del sistema como sea posible. Los valores desconocidos o no disponibles deben definirse con valores razonables (usualmente `0`).

banderas contiene banderas adicionales definidas por la interfaz de programación de secuencias. Puede contener uno o más de los siguientes valores, unidos mediante la operación lógica OR.

Bandera	Descripción
STREAM_URL_STAT_LINK	Para recursos con la habilidad de mantener un enlace con otro recurso (tales como una redirección HTTP Location:, o un enlace simbólico en el sistema de archivos). Esta bandera indica que solo se devolverá información sobre el enlace mismo, no sobre el recurso apuntado por el enlace. Esta bandera es definida en respuesta a llamadas de **lstat()**, **is_link()**, o **filetype()**.
STREAM_URL_STAT_QUIET	Si ésta bandera es definida, su envoltura no debe generar error alguno. Si esta bandera no está definida, usted es responsable por el reporte de errores usando la función **trigger_error()** durante la ejecución del proceso stat en la ruta.

string **dir_readdir** (void)

Este método es llamado en respuesta a **readdir()** y debe devolver una cadena que represente el siguiente nombre de archivo en la ubicación abierta por **dir_opendir()**.

bool **dir_rewinddir** (void)

Este método es llamado en respuesta a **rewinddir()** y debe restablecer la salida generada por **dir_readdir()**. Esto quiere decir, la siguiente llamada a **dir_readdir()** debe devolver la primera entrada en la ubicación devuelta por **dir_opendir()**.

bool **dir_closedir** (void)

Este método es llamado en respuesta a **closedir()**. Debe liberar cualquier recurso que haya sido bloqueado o reservado durante la apertura y uso de la secuencia de directorio.

El siguiente ejemplo implementa un gestor del protocolo var:// que permite el acceso para lectura/escritura a una variable global con nombre, usando las funciones de secuencia de sistema de archivos, como **fread()**. El protocolo var:// implementado a continuación leerá/escribirá datos desde/hacia $GLOBALS["foo"] dada la URL "var://foo".

Ejemplo 1. Una Secuencia para la lectura/escritura de variables globales

```php
<?php

class SecuenciaVariable {
    var $posicion;
    var $nombre_var;

    function stream_open($ruta, $modo, $opciones, &$ruta_abierta)
    {
        $url = parse_url($ruta);
        $this->nombre_var = $url["host"];
        $this->posicion = 0;

        return true;
    }

    function stream_read($conteo)
    {
        $ret = substr($GLOBALS[$this->nombre_var], $this->posicion, $conteo);
        $this->posicion += strlen($ret);
        return $ret;
    }

    function stream_write($datos)
    {
        $izq = substr($GLOBALS[$this->nombre_var], 0, $this->posicion);
        $der = substr($GLOBALS[$this->nombre_var], $this->posicion + strlen($datos));
        $GLOBALS[$this->nombre_var] = $izq . $datos . $der;
        $this->posicion += strlen($datos);
        return strlen($datos);
    }

    function stream_tell()
    {
        return $this->posicion;
    }

    function stream_eof()
    {
        return $this->posicion >= strlen($GLOBALS[$this->nombre_var]);
    }

    function stream_seek($desplazamiento, $partida)
    {
        switch ($partida) {
            case SEEK_SET:
                if ($desplazamiento < strlen($GLOBALS[$this->nombre_var]) && $desplazam
                    $this->posicion = $desplazamiento;
                    return true;
                } else {
                    return false;
                }
                break;

            case SEEK_CUR:
                if ($desplazamiento >= 0) {
                    $this->posicion += $desplazamiento;
                    return true;
                } else {
                    return false;
                }
                break;

            case SEEK_END:
                if (strlen($GLOBALS[$this->nombre_var]) + $desplazamiento >= 0) {
                    $this->posicion = strlen($GLOBALS[$this->nombre_var]) + $desplazam
                    return true;
                } else {
                    return false;
                }
                break;

            default:
```

stream_wrapper_restore

(no version information, might be only in CVS)

stream_wrapper_restore -- Restores a previously unregistered built-in wrapper

Description

bool **stream_wrapper_restore** (string protocol)

Restores a built-in wrapper previously unregistered with **stream_wrapper_unregister()**.

stream_wrapper_unregister

(no version information, might be only in CVS)

stream_wrapper_unregister -- Unregister a URL wrapper

Description

bool **stream_wrapper_unregister** (string protocol)

stream_wrapper_unregister() allows you to disable an already defined stream wrapper. Once the wrapper has been disabled you may override it with a user-defined wrapper using **stream_wrapper_register()** or reenable it later on with **stream_wrapper_restore()**.

CXXI. Funciones de Cadenas

Introducción

Todas estas funciones manipulan cadenas en varias formas. Algunas secciones más especializadas pueden encontrarse en los capítulos sobre expresiones regulares y gestión de URLs.

Para más información sobre el modo en que se comportan las cadenas, especialmente en lo que respecta al uso de comillas sencillas, comillas dobles, y secuencias de escape, vea la entrada Cadenas en la sección Tipos del manual.

Requirimientos

No se necesitan bibliotecas externas para construir esta extensión

---

Instalación

No se necesita ninguna instalación para usar estas funciones, son parte del núcleo de PHP.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

CRYPT_SALT_LENGTH integer

CRYPT_STD_DES integer

CRYPT_EXT_DES integer

CRYPT_MD5 integer

CRYPT_BLOWFISH integer

HTML_SPECIALCHARS (integer)

HTML_ENTITIES (integer)

ENT_COMPAT (integer)

ENT_QUOTES (integer)

ENT_NOQUOTES (integer)

CHAR_MAX (integer)

LC_CTYPE (integer)

LC_NUMERIC (integer)

LC_TIME (integer)

LC_COLLATE (integer)

LC_MONETARY (integer)

LC_ALL (integer)

LC_MESSAGES (integer)

STR_PAD_LEFT (integer)

STR_PAD_RIGHT (integer)

STR_PAD_BOTH (integer)

Ver también

Para consultar sobre funciones de gestión y manipulación de cadenas incluso más poderosas, eche un vistazo a las funciones de expresiones regulares POSIX y las funciones de expresiones regulares compatibles con Perl.

Tabla de contenidos

sscanf -- Trocea la entrada desde una cadena según un formato dado
str_ireplace -- Versión insensible a mayúsculas y minúsculas de **str_replace()**.
str_pad -- Rellena una cadena con otra hasta una longitud dada
str_repeat -- Repite una cadena
str_replace -- Sustitiye todas las apariciones de la aguja en el pajar por la cadena
str_rot13 -- Realizar la transformación rot13 sobre una cadena
str_shuffle -- Reordena aleatoriamente una cadena
str_split -- Convertir una cadena en una matriz
str_word_count -- Devolver información sobre las palabras usadas en una cadena
strcasecmp -- Comparación de cadenas insensible a mayúsculas y minúsculas y segura en modo binario
strchr -- Encuentra la primera aparición de un caracter
strcmp -- Comparación de cadenas con seguridad binaria
strcoll -- Comparación de cadenas basada en la localidad
strcspn -- Encuentra la longitud del elemento inicial que no coincide con la máscara
strip_tags -- Elimina marcas HTML y PHP de una cadena
stripcslashes -- Desmarca la cadena marcada con **addcslashes()**
stripos -- Encontrar la posición de la primera ocurrencia de una cadena, insensible a mayúsculas y minúsculas
stripslashes -- Desmarca la cadena marcada con **addslashes()**
stristr -- **strstr()** sin tener en cuenta mayúsculas o minúsculas
strlen -- Obtiene la longitud de la cadena
strnatcasecmp -- Comparación de cadenas insensible a mayúsculas y minúsculas usando un algoritmo de "orden natural"
strnatcmp -- Compara cadenas usando un algoritmo de "orden natural"
strncasecmp -- Comparación de los primeros n caracteres de cadenas, segura con material binario e insensible a mayúsculas y minúsculas
strncmp -- Comparación de los n primeros caracteres de cadenas, con seguridad binaria
strpbrk -- Busca una cadena por cualquiera de los elementos de un conjunto de caracteres
strpos -- Encuentra la posición de la primera aparición de una cadena
strrchr -- Encuentra la última aparición de un caracter en una cadena
strrev -- Invierte una cadena
strripos -- Encontrar la posición de la última ocurrencia de una cadena en otra, insensible a mayúsculas y minúsculas
strrpos -- Encuentra la posición de la última aparición de un caracter en una cadena
strspn -- Encuentra la longitud del segmento inicial que coincide con la máscara
strstr -- Encuentra la primera aparición de una cadena
strtok -- Divide una cadena en elementos
strtolower -- Pasa a minúsculas una cadena
strtoupper -- Pasa a mayúsculas una cadena
strtr -- Traduce ciertos caracteres
substr_compare -- Comparación de 2 cadenas, segura con material binario, opcionalmente insensible a mayúsculas y minúsculas, a partir de un desplazamiento, y hasta un número límite de caracteres
substr_count -- Cuenta el número de apariciones de la subcadena
substr_replace -- Sustituye texto en una parte de una cadena
substr -- Devuelve parte de una cadena
trim -- Elimina espacios del principio y final de una cadena
ucfirst -- Pasar a mayúsculas el primer caracter de una cadena
ucwords -- Pone en mayúsculas el primer caracter de cada palabra de una cadena
vfprintf -- Write a formatted string to a stream
vprintf -- Imprimir una cadena con formato
vsprintf -- Devuelve una cadena con formato
wordwrap -- Corta una cadena en un número dado de caracteres usando un caracter de ruptura de cadenas.

AddCSlashes

(PHP 4 , PHP 5)

AddCSlashes -- Marca una cadena con barras al estilo del C

Descripción

string **addcslashes** (string cad, string listcar)

Devuelve una cadena con barras invertidas antes de los caracteres listados en el parámetro *listcar*. También marca *\n*, *\r* etc. Al estilo del C, los caracteres con códico ASCII inferior a 32 y superior a 126 son convertidos a representación octal. Tenga cuidado cuando marque caracteres alfanuméricos. Puede especificar un rango en *listcar* como el "\0..\37", que marcaría todos los caracteres con código ASCII entre 0 y 31.

Ejemplo 1. Ejemplo de addcslashes()

```
$tradformado = addcslashes ($no_transf, "\0..\37!@\177..\377");
```

Nota: Añadida en PHP4b3-dev.

Vea también **stripcslashes()**, **stripslashes()**, **htmlspecialchars()**, **htmlspecialchars()**, y **quotemeta()**.

AddSlashes

(PHP 3, PHP 4 , PHP 5)

AddSlashes -- Marca una cadena con barras

Descripción

string **addslashes** (string cad)

Devuelve una cadena con barras invertidas frente a los caracteres que necesitan marcarse en consultas de bases de datos, etc. Estos son la comilla simple (′), comilla doble (″), barra invertida (\) y NUL (el byte nulo).

Vea también **stripslashes()**, **htmlspecialchars()**, y **quotemeta()**.

bin2hex

(PHP 3>= 3.0.9, PHP 4 , PHP 5)

bin2hex -- Convierte datos binarios en su representación hexadecimal

Descripción

string **bin2hex** (string cad)

Devuelve una cadena ASCII que contiene la representación hexadecimal de *cad*. La conversión se

realiza byte a byte, con los 4 bits superiores primero.

chop

(PHP 3, PHP 4 , PHP 5)

chop -- Elimina espacios sobrantes al final

Descripción

string **chop** (string cad)

Devuelve la cadena argumento sin los espacios sobrantes, incluyendo los saltos de línea.
Ejemplo 1. Ejemplo de chop()

```
$recortada = chop ($linea);
```

Vea también **trim()**.

chr

(PHP 3, PHP 4 , PHP 5)

chr -- Devuelve un caracter específico

Descripción

string **chr** (int ascii)

Devuelve una cadena de un caracter que congiene el caracter especificado por *ascii*.
Ejemplo 1. Ejemplo de chr()

```
$cad .= chr (27); /* añade un caracter de escape al final de $cad */

/* A veces esto es más útil */

$cad = sprintf ("La cadena termina en escape: %c", 27);
```

Esta función complementa a **ord()**. Vea también **sprintf()** con una cadena de formato *%c*.

chunk_split

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

chunk_split -- Divide una cadena en trozos más pequeños

Descripción

string **chunk_split** (string cadena [, int tamatrozo [, string final]])

Se puede utilizar para trocear una cadena en pedazos más pequeños, lo que es útil, p.ej., para convertir la salida de la función base64_encode a la semántica del RFC 2045. Inserta la cadena *final* cada *tamatrozo* (por defecto vale 76) caracteres. Devuelve la nueva cadena y deja intacta la original.
Ejemplo 1. Ejemplo de chunk_split()

```
# formatear $datos usando la semántica del RFC 2045

$nueva_cad = chunk_split (base64_encode($datos));
```

Esta función es notablemente más rápida que **ereg_replace()**.

> **Nota:** Esta función se añadió en la 3.0.6.

convert_cyr_string

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

convert_cyr_string -- Convierte de un juego de caracteres Cirílico a otro

Descripción

string **convert_cyr_string** (string cad, string desde, string hasta)

Esta función convierte la cadena dada de un juego de caracteres Cirílico a otro. Los argumentos *desde* y *hasta* son caracteres sencillos que representan los juegos de caracteres Cirílicos fuente y destino. Los tipos soportados son:

- k - koi8-r
- w - windows-1251
- i - iso8859-5
- a - x-cp866
- d - x-cp866
- m - x-mac-cyrillic

convert_uudecode

(PHP 5)

convert_uudecode -- Descifra una cadena codificada mediante uuencode

Descripción

string **convert_uudecode** (string datos)

convert_uudecode() descifra una cadena codificada mediante uuencode.

Ejemplo 1. Ejemplo de convert_uudecode()

```
<?php
/* Puede imaginarse lo que esto imprime? :) */
echo convert_uudecode("+22!L;W9E(%!(4\"$`\n`");
?>
```

Vea también **convert_uuencode()**.

convert_uuencode

(PHP 5)

convert_uuencode -- Codifica, mediante uuencode, una cadena

Descripción

string **convert_uuencode** (string datos)

convert_uuencode() codifica una cadena usando el algoritmo uuencode.

Uuencode traduce todas las cadenas (incluyendo las binarias) a secuencias de caracteres imprimibles, haciéndolas seguras para transmisiones de red. Los datos codificados de esta forma son aproximadamente 35% más largos que el original.

Ejemplo 1. Ejemplo de convert_uuencode()

```
<?php
$una_cadena = "prueba\ntexto texto\r\n";

echo convert_uuencode($una_cadena);
?>
```

Vea también **convert_uudecode()** y **base64_encode()**.

count_chars

(PHP 4 , PHP 5)

count_chars -- Devuelve información sobre los caracteres usados en una cadena

Descripción

mixed **count_chars** (string cadena [, modo])

Cuenta el número de apariciones de cada valor de byte (0..255) en *cadena* y lo devuelve de varias maneras. El parámetro opcional *modo* vale por defecto 0. Dependiendo de *modo*, **count_chars()** puede devolver:

- 0 - una matriz con el valor del byte como clave y la frecuencia de cada uno como valor.
- 1 - como el 0, pero listando únicamente los valores de byte con frecuencia superior a cero.
- 2 - como el 0, pero listando únicamente los valores de byte con frecuencia igual a 0.
- 3 - se devuelve una cadena que contiene todos los valores de byte utilizados.
- 4 - se devuelve una cadena que contiene todos los valores de byte no utilizados.

Nota: Esta función se añadió en el PHP 4.0.

crc32

(PHP 4 >= 4.0.1, PHP 5)

crc32 -- Calcula el polinomio crc32 de una cadena

Descripción

int **crc32** (string cad)

Genera el polinomio de comprobación de reduncancia cíclica de 32 bits de *cad*. Se suele utilizar para validad la integridad de los datos transmitidos.

Vea también: **md5()**

crypt

(PHP 3, PHP 4 , PHP 5)

crypt -- Encripta una cadena mediante DES

Descripción

string **crypt** (string cad [, string semilla])

crypt() encriptará una cadena utilizando el método estándar de encriptación del Unix DES. Los argumentos son una cadena a encriptar y una cadena semilla de 2 caracteres en la que basar la encriptación. Vea la página de manual de Unix sobre crypt para más información.

Si el argumento de semilla no se proporciona, será generado aleatoriamente por el PHP.

Algunos sistemas operativos soportan más de un tipo de encriptación. De hecho, algunas veces la encriptación estándar DES es sustituída por un algoritmo de encriptación basado en MD5. El tipo de encriptación es disparado por el argumento semilla. En tiempo de instalación, el PHP determina la capacidad de la función de encriptación y aceptará semillas para otros tipos de encriptación. Si no se proporciona la semilla, el PHP intentará generar una semilla estándar DES de 2 caraceres por defecto, excepto si el tipo de encriptación estándar del sistema es el MD5, en cuyo caso se generará una semilla aleatoria compatible con MD5. El PHP fija una constante llamada CRYPT_SALT_LENGTH que le especifica si su sistema soporta una semilla de 2 caracteres o si se debe usar la semilla de 12 caracteres del NDS.

La función estándar de encriptación **crypt()** contiene la semilla como los dos primeros caracteres de la salida.

En los sistemas en los que la función crypt() soporta múltiples tipos de encriptación, las siguienes constantes son fijadas a 0 ó 1 dependiendo de si está disponible el tipo dado:

- CRYPT_STD_DES - Encriptación DES estándar con semilla de 2 caracteres
- CRYPT_EXT_DES - Encriptación DES extendida con semilla de 9 caracteres

- CRYPT_MD5 - Encriptación MD5 con semilla de 12 caracteres y comenzando por 1
- CRYPT_BLOWFISH - Encriptación DES extendida con semilla de 16 caracteres y comenzando por 2

No hay función de desencriptado porque **crypt()** utiliza un algoritmo de una sola vía.

Vea también: **md5()**.

echo

(PHP 3, PHP 4, PHP 5)

echo -- Da salida a una o más cadenas

Descripción

echo (string arg1 [, string argn...])

Da salida a todos sus parámetros.

echo() no es realmente una función (es una sentencia del lenguaje) de modo que no se requiere el uso de los paréntesis.

Ejemplo 1. Ejemplo de echo()

```
echo "Hola Mundo";

echo "Esto se extiende
por varias líneas. Los saltos de línea
también se envían";

echo "Esto se extiende\npor varias líneas. Los saltos de línea\ntambién se envían";
```

> **Nota:** De hecho, si desea pasar más de un parámetro a echo no debe encerrarlos entre paréntesis.

Vea también: **print()**, **printf()**, y **flush()**.

explode

(PHP 3, PHP 4 , PHP 5)

explode -- Divide una cadena por otra

Descripción

array **explode** (string separador, string cadena [, int limite])

Devuelve una matriz de cadenas, cada una de las cuales es una subcadena de *cadena* formada mediante su división en las fronteras marcadas por la cadena *separador*. Si se especifica *limite*, la matriz devuelta contendrá un máximo de *limite* elementos con el último conteniendo el resto de la *cadena*.

Ejemplo 1. Ejemplo de explode()

```
$pizza = "trozo1 trozo2 trozo3 trozo4 trozo5 trozo6";
$trozos = explode (" ", $pizza);
```

Vea también **split()** e **implode()**.

fprintf

(PHP 5)

fprintf -- Escribir una cadena con formato a una secuencia

Descripción

int **fprintf** (resource gestor, string formato [, mixed args [, mixed ...]])

Escribe una cadena producida de acuerdo a *formato* al recurso de secuencia especificado por *gestor*. *formato* se describe en la documentación para **sprintf()**.

Devuelve la longitud de la cadena impresa.

Vea también: **printf()**, **sprintf()**, **sscanf()**, **fscanf()**, **vsprintf()**, y **number_format()**.

Ejemplos

Ejemplo 1. fprintf(): enteros con relleno de ceros

```
<?php
if (!($da = fopen('fecha.txt', 'w')))
    return;

fprintf($da, "%04d-%02d-%02d", $anyo, $mes, $dia);
// escribe la fecha en formato ISO a fecha.txt
?>
```

Ejemplo 2. fprintf(): dando formato a valores monetarios

```
<?php
if (!($da = fopen('monetario.txt', 'w')))
    return;

$dinero1 = 68.75;
$dinero2 = 54.35;
$dinero = $dinero1 + $dinero2;
// echo $dinero producira la salida "123.1";
$long = fprintf($da, '%01.2f', $dinero);
// escribe "123.10" en monetario.txt

echo "se escribieron $long bytes a monetario.txt";
// usar el valor de retorno para determinar cuantos bytes se escribieron
?>
```

get_html_translation_table

(PHP 4 , PHP 5)

get_html_translation_table -- Devuelve la tabla de traducción utilizada por **htmlspecialchars()** y **htmlentities()**

Descripción

string **get_html_translation_table** (int tabla)

get_html_translation_table() devolverá la tabla de traducción que se usa internamente para **htmlspecialchars()** y **htmlentities()**. Hay dos nuevas definiciones (*HTML_ENTITIES*, *HTML_SPECIALCHARS*) que le permiten especificar la tabla deseada.

Ejemplo 1. Ejemplo de Tabla de Traducción

```
$trad = get_html_translation_table (HTML_ENTITIES);
$cad = "Hallo & <Frau> & KrÃƒÂ¤mer";
$codif = strtr ($cad, $trad);
```

La variable *$codif* contendrá ahora: "Hallo &**<amp>**; &**<lt>**;Frau&**<gt>**; &**<amp>**; Kr&**<auml>**;mer".

Lo interesante es usar la función **array_flip()** para cambiar la dirección de la traducción.

```
$trad = array_flip ($trad);
$original = strtr ($cad, $trad);
```

El contenido de *$original* sería: "Hallo & <Frau> & KrÃƒÂ¤mer".

> **Nota:** Esta función fue añadida en PHP 4.0.

Vea también: **htmlspecialchars()**, **htmlentities()**, **strtr()**, y **array_flip()**.

hebrev

(PHP 3, PHP 4 , PHP 5)

hebrev -- Convierte Hebreo lógico a texto visual

Descripción

string **hebrev** (string texto_hebreo [, int max_cars_por_linea])

El parámetro opcional *max_cars_por_linea* indica el máximo número de caracteres que se emitirán por línea. La función intenta evitar cortar palabras.

Vea también **hebrevc()**

hebrevc

(PHP 3, PHP 4 , PHP 5)

hebrevc -- Convierte Hebreo lógico a texto visual con conversión de saltos de línea

Descripción

string **hebrevc** (string texto_hebreo [, int max_cars_por_linea])

Esta funcion es similar a **hebrev()** con la diferencia que convierte las nuevas líneas (\n) a "
\n".

El parámetro opcional *max_cars_por_linea* indica el máximo número de caracteres que se emitirán por línea. La función intenta evitar cortar palabras.

Vea también **hebrev()**

html_entity_decode

(PHP 4 >= 4.3.0, PHP 5)

html_entity_decode -- Convertir todas las entidades HTML a sus caracteres correspondientes

Descripción

string **html_entity_decode** (string cadena [, int estilo_comillas [, string juego_caracteres]])

html_entity_decode() es el opuesto de **htmlentities()** en el sentido en que convierte todas las entidades HTML a sus caracteres correspondientes en la *cadena* dada.

El segundo parámetro opcional *estilo_comillas* le permite definir lo que debe hacerse con las comillas 'sencillas' y "dobles". Recibe una de tres constantes posibles, siendo el valor por defecto **ENT_COMPAT**:

Tabla 1. Constantes disponibles para *estilo_comillas*

Nombre de constante	Descripción
ENT_COMPAT	Convierte las comillas dobles y deja intactas las comillas sencillas.
ENT_QUOTES	Convierte tanto comillas dobles como sencillas.
ENT_NOQUOTES	No convierte ni las comillas dobles ni las sencillas.

El juego de caracteres ISO-8859-1 es usado como valor predeterminado para el tercer argumento opcional *juego_caracteres*. Éste define el juego de caracteres usado en la conversión.

Los siguientes juegos de caracteres son soportados a partir de PHP 4.3.0.

Tabla 2. Juegos de caracteres soportados

Juego de caracteres	Aliases	Descripción
ISO-8859-1	ISO8859-1	Europeo Occidental, Latin-1
ISO-8859-15	ISO8859-15	Europeo Occidental, Latin-9. Añade el signo de Euro, y letras del Francés y Finlandés que hacían falta en Latin-1(ISO-8859-1).
UTF-8		Multi-byte Unicode de 8-bits compatible con ASCII.
cp866	ibm866, 866	Juego de caracteres cirílicos específico de DOS. Este juego de caracteres está soportado en 4.3.2.
cp1251	Windows-1251, win-1251, 1251	Juego de caracteres cirílicos específico de Windows. Este juego de caracteres está soportado en 4.3.2.

Juego de caracteres	Aliases	Descripción
cp1252	Windows-1252, 1252	Juego de caracteres específico de Windows para Europa Occidental.
KOI8-R	koi8-ru, koi8r	Ruso. Este juego de caracteres está soportado en 4.3.2.
BIG5	950	Chino Tradicional, usado principalmente en Taiwán.
GB2312	936	Chino Simplificado, juego de caracteres estándar nacional.
BIG5-HKSCS		Big5 con extensiones de Hong Kong, Chino Tradicional.
Shift_JIS	SJIS, 932	Japonés
EUC-JP	EUCJP	Japonés

Nota: Cualquier otro juego de caracteres no es reconocido y en su lugar se utilizará ISO-8859-1.

Ejemplo 1. Decodificación de entidades HTML

```
<?php
$orig = "I'll \"walk\" the <b>dog</b> now";

$a = htmlentities($orig);

$b = html_entity_decode($a);

echo $a; // I'll &quot;walk&quot; the &lt;b&gt;dog&lt;/b&gt; now

echo $b; // I'll "walk" the <b>dog</b> now


// Usuarios de una version anterior a 4.3.0 de PHP, pueden hacer esto:
function unhtmlentities($cadena)
{
    $trans_tbl = get_html_translation_table(HTML_ENTITIES);
    $trans_tbl = array_flip($trans_tbl);
    return strtr($cadena, $trans_tbl);
}

$c = unhtmlentities($a);

echo $c; // I'll "walk" the <b>dog</b> now

?>
```

Nota: Puede que se pregunte por qué trim(html_entity_decode(' ')); no reduce la cadena a una cadena vacía, esto es porque la entidad ' ' no es el código ASCII 32 (el cual es eliminado por **trim()**) sino el código ASCII 160 (0xa0) en el juego de caracteres por defecto, ISO 8859-1.

Vea también **htmlentities()**, **htmlspecialchars()**, **get_html_translation_table()**, y **urldecode()**.

htmlentities

(PHP 3, PHP 4 , PHP 5)

htmlentities -- Convierte todos los caracteres aplicables a entidades HTML

Descripción

string **htmlentities** (string cadena)

Esta función es del todo idéntica a **htmlspecialchars()**, excepto que traduce todos los caracteres que tienen ñalente como entidad HTML.

Actualmente se utiliza el juego de caracteres ISO-8859-1.

Vea también **htmlspecialchars()** y **nl2br()**.

htmlspecialchars

(PHP 3, PHP 4 , PHP 5)

htmlspecialchars -- Convierte caracteres especiales a entidades HTML

Descripción

string **htmlspecialchars** (string cadena)

Ciertos caracteres tienen significados especiales en HTML, y deben ser representados por entidades HTML si se desea preservar su significado. Esta función devuelve una cadena con dichas conversiones realizadas.

Esta función es útil para evitar que el texo entrado por el usuario contenga marcas HTML, como ocurre en aplicaciones de foros o libros de visita.

Actualmente, las traducciones hechas son:

- '&' (ampersand) se convierte en '&'
- '"' (doble comilla) se convierte en '"'
- '<' (menor que) se convierte en '<'
- '>' (mayor que) se convierte en '>'

Nótese que esta función no traduce nada más que lo mostrado más arriba. Para una traducción de entidades completa, vea **htmlentities()**.

Vea también **htmlentities()** y **nl2br()**.

implode

(PHP 3, PHP 4 , PHP 5)

implode -- Unir elementos de una matriz mediante una cadena

Descripción

string **implode** (string cola, array piezas)

Devuelve una cadena que contiene una representación de todos los elementos de la matriz en el mismo orden, pero con la cadena *cola* en medio de los mismos.

Ejemplo 1. Ejemplo de implode()

```
$separada_dospuntos = implode (":", $matrizay);
```

Vea también **explode()**, **join()**, y **split()**.

join

(PHP 3, PHP 4 , PHP 5)

join -- Une elementos de una tabla mediante una cadena

Descripción

string **join** (string cola, array piezas)

join() es un alias para **implode()**, y es idéntica en todo.

Vea también **explode()**, **implode()**, y **split()**.

levenshtein

(PHP 3>= 3.0.17, PHP 4 >= 4.0.1, PHP 5)

levenshtein -- Calcula la distancia Levenshtein entre dos cadenas

Descripción

int **levenshtein** (string cad1, string cad2)

Esta función devuelve la distancia Levenshtein entre las dos cadenas argumento, ó -1 si alguna de las cadenas tiene más de 255 caracteres.

La distancia Levenshtein se define como el mínimo número de caracteres que se tienen que sustituir, insertar o borrar para transformar *cad1* en *cad2*. La complejidad del algoritmo es *O(m*n)*, donde *n* y *m* son las longitudes de *cad1* y *cad2* (bastante bueno si se la compara con **similar_text()**, que es O (max(n,m)**3), pero aún es cara).

Vea también **soundex()**, **similar_text()** y **metaphone()**.

localeconv

(PHP 4 >= 4.0.5, PHP 5)

localeconv -- Obtener información sobre el formato numérico

Descripción

array **localeconv** (void)

Devuelve una matriz asociativa que contiene información de los formatos numérico y monetario,

localizados.

localeconv() devuelve información basada en la localidad actual, tal y como haya sido definida mediante **setlocale()**. La matriz asociativa que devuelve contiene los siguientes campos:

Elemento de la matriz	Descripción
decimal_point	Caracter de punto decimal
thousands_sep	Separador de miles
grouping	Matriz que contiene agrupaciones numéricas
int_curr_symbol	Símbolo internacional de moneda (i.e. USD)
currency_symbol	Símbolo local de moneda (i.e. $)
mon_decimal_point	Caracter de punto decimal monetario
mon_thousands_sep	Separador de miles monetario
mon_grouping	Matriz que contiene agrupaciones de moneda
positive_sign	Signo para valores positivos
negative_sign	Signo para valores negativos
int_frac_digits	Dígitos fraccionarios internacionales
frac_digits	Dígitos fraccionarios locales
p_cs_precedes	**TRUE** si currency_symbol precede un valor positivo, **FALSE** si lo sucede
p_sep_by_space	**TRUE** si un espacio separa currency_symbol de un valor positivo, **FALSE** de lo contrario
n_cs_precedes	**TRUE** si currency_symbol precede un valor negativo, **FALSE** si lo sucede
n_sep_by_space	**TRUE** si un espacio separa currency_symbol de un valor negativo, **FALSE** de lo contrario
p_sign_posn	0 Paréntesis rodean la cantidad y currency_symbol 1 La cadena de signo precede la cantidad y currency_symbol 2 La cadena de signo sucede la cantidad y currency_symbol 3 La cadena de signo precede inmediatamente currency_symbol 4 La cadena de signo sucede inmediatamente currency_symbol
n_sign_posn	0 Paréntesis rodean la cantidad y currency_symbol 1 La cadena de signo precede la cantidad y currency_symbol 2 La cadena de signo sucede la cantidad y currency_symbol 3 La cadena de signo precede inmediatamente currency_symbol 4 La cadena de signo sucede inmediatamente currency_symbol

Los campos de agrupamiento contienen matrices que definen el modo en que los números deben ser agrupados. Por ejemplo, el campo de agrupamiento para la localidad en_US debería contener una matriz de 2 elementos con los valores 3 y 3. Entre más alto sea el índice dentro de la matriz, lo más lejos hacia la izquierda se encuentra el agrupamiento. Si un elemento de la matriz es igual a CHAR_MAX, no se realiza ningún agrupamiento posterior. Si un elemento de la matriz es igual a 0, el elemento anterior deberá ser usado.

Ejemplo 1. Ejemplo de localeconv()

```
<?php
setlocale(LC_ALL, "en_US");

$info_locale = localeconv();

echo "<pre>\n";
echo "-------------------------------------------------\n";
echo " Informaci&oacute;n monetaria para la localidad actual:  \n";
echo "-------------------------------------------------\n\n";

echo "int_curr_symbol:   {$info_locale["int_curr_symbol"]}\n";
echo "currency_symbol:   {$info_locale["currency_symbol"]}\n";
echo "mon_decimal_point: {$info_locale["mon_decimal_point"]}\n";
echo "mon_thousands_sep: {$info_locale["mon_thousands_sep"]}\n";
echo "positive_sign:     {$info_locale["positive_sign"]}\n";
echo "negative_sign:     {$info_locale["negative_sign"]}\n";
echo "int_frac_digits:   {$info_locale["int_frac_digits"]}\n";
echo "frac_digits:       {$info_locale["frac_digits"]}\n";
echo "p_cs_precedes:     {$info_locale["p_cs_precedes"]}\n";
echo "p_sep_by_space:    {$info_locale["p_sep_by_space"]}\n";
echo "n_cs_precedes:     {$info_locale["n_cs_precedes"]}\n";
echo "n_sep_by_space:    {$info_locale["n_sep_by_space"]}\n";
echo "p_sign_posn:       {$info_locale["p_sign_posn"]}\n";
echo "n_sign_posn:       {$info_locale["n_sign_posn"]}\n";
echo "</pre>\n";
?>
```

La constante CHAR_MAX también se define para el uso mencionado anteriormente.

Vea también **setlocale()**.

ltrim

(PHP 3, PHP 4 , PHP 5)

ltrim -- Elimina el espacio en blanco del principio de una cadena

Descripción

string **ltrim** (string cad)

Esta función elimina el espacio en blanco del principio de una cadena y devuelve la cadena resultante. Los caracteres de espacio que elimina realmente son: "\n", "\r", "\t", "\v", "\0", y el espacio en sí.

Vea también **chop()** y **trim()**.

md5_file

(PHP 4 >= 4.2.0, PHP 5)

md5_file -- Calcula el resumen criptográfico md5 de un nombre de archivo dado

Descripción

string **md5_file** (string nombre_archivo [, bool salida_primitiva])

Calcula el resumen criptográfico MD5 del *nombre_archivo* especificado, usando el Algoritmo de

Resumen de Mensajes MD5 de RSA Data Security, Inc., y devuelve ese resumen. El resumen criptográfico es un número hexadecimal de 32 caracteres. Si el valor opcional *salida_primitiva* es **TRUE**, entonces el resumen md5 es retornado en cambio en formato binario primitivo con una longitud de 16.

> **Nota:** El parámetro opcional *salida_primitiva* fue añadido en PHP 5.0.0 y tiene un valor predeterminado de **FALSE**

Esta función tiene el mismo propósito que la utilidad de línea de comandos md5sum.

Vea también **md5()**, **crc32()**, y **sha1_file()**.

md5

(PHP 3, PHP 4 , PHP 5)

md5 -- Calcula el hash md5 de una cadena

Descripción

string **md5** (string cad)

Calcula el hash (extracto) MD5 de *cad* usaneo el Algoritmo de Resumen de Mensajes MD5 de RSA Data Security, Inc. .

Vea también: **crc32()**

metaphone

(PHP 4 , PHP 5)

metaphone -- Calcula la clave "metáfona" de una cadena

Descripción

string **metaphone** (string cad)

Calcula la clave "metáfona" de *cad*.

Similarmente a **soundex()**, metaphone crea la misma clave para palabras que suenan parecidas. Es más precisa que la función **soundex()**, pues conoce las reglas básicas de la pronunciación del Inglés. Las claves metafónicas generadas son de longitud variable.

Metaphone fue desarrollado por Lawrence Philips <lphilips@verity.com>. Se describe en ["Practical Algorithms for Programmers", Binstock & Rex, Addison Wesley, 1995].

> **Nota:** Esta función se añadió en PHP 4.0.

money_format

(PHP 4 >= 4.3.0, PHP 5)

money_format -- Da formato a un número como una cadena de moneda

Descripción

string **money_format** (string formato, float numero)

money_format() devuelve una versión con formato de *numero*. Esta función envuelve la función de la biblioteca de C **strfmon()**, con la diferencia de que ésta implementación convierte solo un número a la vez.

> **Nota:** La función **money_format()** está definida solo si el sistema tiene la capacidad de llamar a strfmon. Por ejemplo, Windows no tiene soporte para esta función, así que **money_format()** no se encuentra definida en Windows.

La especificación de formato consiste de la siguiente secuencia:

- un caracter *%*
- banderas opcionales
- un ancho de campo opcional
- precisión de izquierda opcional
- precisión de derecha opcional
- un caracter de conversión, requerido

Banderas. Una o más de las siguientes banderas opcionales pueden ser usadas:

=f

> El caracter = seguido de un caracter (byte sencillo) *f* que será usado como el caracter de relleno numérico. Es caracter de relleno predeterminado es el espacio.

^

> Deshabilitar el uso de caracteres de agrupamiento (tal y como estén definidos según la localidad actual).

+ o *(*

> Especifica el estilo de formato para números positivos y negativos. Si se usa +, los ñalentes de + y - en la localidad actual serán usados. Si se usa *(*, las cantidades negativas estarán ubicadas entre paréntesis. Si no se da especificación alguna, el valor por defecto es +.

!

Elimina el símbolo de moneda de la cadena de salida.

-

Si se encuentra presente, hará que todos los campos estén justificados a izquierda (con relleno a la derecha), en contraste al comportamiento predeterminado que hace que los campos estén justificados a derecha (con relleno a la izquierda).

Ancho de campo.

w

Una cadena de dígito decimal que especifica un ancho de campo mínimo. El campo será justificado a derecha a menos que la bandera - sea usada. El valor predeterminado es 0 (cero).

Precisión de izquierda.

#n

El número máximo de dígitos (*n*) a esperar a la izquerda del caracter decimal (p. ej. el punto decimal). Usualmente se usa para mantener la salida con formato alineada en las mismas columnas, usando el caracter de relleno si el número de dígitos es menor que *n*. Si el número de dígitos real es mayor que *n*, entonces ésta especificación es ignorada.

Si no se ha suprimido el agrupamiento usando la bandera ^, los separadores de agrupamiento serán insertados antes de que los caracteres de relleno (si los hay) sean agregados. Los separadores de agrupamiento no serán aplicados sobre los caracteres de relleno, aun si el caracter de relleno es un dígito.

Para asegurar el alineamiento, cualquier caracter que aparezca antes o después del número en la salida con formato, tales como los símbolos de moneda o de signo, son rellenados en tanto sea necesario con caracteres de espacio para hacer que sus formatos positivo y negativo tengan una misma longitud.

Precisión de derecha .

.p

Un punto seguido del número de dígitos (*p*) después del caracter decimal. Si el valor de *p* es 0 (cero), el caracter decimal y los dígitos a su derecha serán omitidos. Si no se incluye ninguna precisión de derecha, el valor predeterminado será determinado por la localidad en uso. La cantidad a la cual se está dando formato es redondeada al número de dígitos especificado antes del formato.

Caracteres de conversión .

i

El número recibe formato de acuerdo al formato de moneda internacional de la localidad (p.ej. para la localidad de USA: USD 1,234.56).

n

El número recibe formato de acuerdo al formato de moneda nacional de la localidad (p.ej.

para la localidad de_DE: DM1.234,56).

%

Devuelve el caracter *%*.

Nota: La categoría **`LC_MONETARY`** de los parámetros de la localidad, afecta el comportamiento de esta función. Use **setlocale()** para establecer la localidad por defecto apropiada antes de usar esta función.

Los caracteres antes y después de la cadena de formato serán devueltos sin modificaciones.

Ejemplo 1. Ejemplo de money_format()

Usaremos diferentes localidades y especificaciones de formato para ilustrar el uso de esta función.

```
<?php

$numero = 1234.56;

// imprimamos el formato internacional para la localidad en_US
setlocale(LC_MONETARY, 'en_US');
echo money_format('%i', $numero) . "\n";
// USD 1,234.56

// Formato nacional italiano con 2 decimales
setlocale(LC_MONETARY, 'it_IT');
echo money_format('%.2n', $numero) . "\n";
// L. 1.234,56

// Uso de un numero negativo
$numero = -1234.5672;

// Formato nacional de US, usando () para numeros negativos
// y 10 digitos para la precision de derecha
setlocale(LC_MONETARY, 'en_US');
echo money_format('%(#10n', $numero) . "\n";
// ($        1,234.57)

// Un formato similar al anterior, agregando el uso de 2 digitos de
// precision de derecha y '*' como caracter de relleno
echo money_format('%=*(#10.2n', $numero) . "\n";
// ($********1,234.57)

// Justifiquemos a izquierda, con 14 posiciones de ancho, 8 digitos de
// precision de izquierda, 2 de precision de derecha, sin caracter de
// agrupamiento y usando el formato internacional de la localidad
// de_DE.
setlocale(LC_MONETARY, 'de_DE');
echo money_format('%=*^-14#8.2i', 1234.56) . "\n";
// DEM 1234,56****

// Agreguemos un poco de informacion antes y despues de la especificacion
// de conversion
setlocale(LC_MONETARY, 'en_GB');
$fmt = 'El valor final es %i (despues de un descuento de 10%%)';
echo money_format($fmt, 1234.56) . "\n";
// El valor final es GBP 1,234.56 (despues de un descuento de 10%)

?>
```

Vea también: **setlocale()**, **number_format()**,**sprintf()**, **printf()** y **sscanf()**.

nl_langinfo

(PHP 4 >= 4.1.0, PHP 5)

nl_langinfo -- Consultar información sobre el lenguaje y la localidad

Descripción

string **nl_langinfo** (int item)

La función **nl_langinfo()** es usada para acceder a elementos individuales de las categorías de localidad. A diferencia de **localeconv()**, que devuelve todos los elementos, **nl_langinfo()** le permite seleccionar cualquier elemento específico.

Si *item* no es válido, se devolverá **`FALSE`**.

item puede ser un valor entero del elemento o el nombre de constante del elemento. La siguiente es una lista de nombres de constante para *item* que pueden ser usados y su descripción. Puede que algunas de éstas constantes no estén definidas o no contengan valor alguno para ciertas localidades.

Tabla 1. Constantes de nl_langinfo

Constante	Descripción
Constantes de la Categoría LC_TIME	
ABDAY_(1-7)	Nombre abreviado del n-ésimo día de la semana.
DAY_(1-7)	Nombre abreviado del n-ésimo día de la semana (DAY_1 = Domingo).
ABMON_(1-12)	Nombre abreviado del mes n-ésimo del año.
MON_(1-12)	Nombre del mes n-ésimo del año.
AM_STR	Cadena del ante-meridiano.
PM_STR	Cadena del post-meridiano.

Constante	Descripción
D_T_FMT	Cadena que puede ser usada como cadena de formato para **strftime()**, para representar la hora y fecha.
D_FMT	Cadena que puede ser usada como cadena de formato para **strftime()**, para representar la fecha.
T_FMT	Cadena que puede ser usada como la cadena de formato para **strftime()**, para representar la hora.
T_FMT_AMPM	Cadena que puede ser usada como cadena de formato para **strftime()** para representar la hora en formato 12-horas con ante/post meridiano.
ERA	Era alterna.
ERA_YEAR	Año en formato de era alterna.
ERA_D_T_FMT	Fecha y hora en formato de era alterna (la cadena puede ser usada en **strftime()**).
ERA_D_FMT	Fecha en formato de era alterna (la cadena puede ser usada en **strftime()**).

Constante	Descripción
ERA_T_FMT	Hora en formato de era alterna (la cadena puede ser usada en **strftime()**).
Constantes de la Categoría LC_MONETARY	
INT_CURR_SYMBOL	Símbolo de moneda internacional.
CURRENCY_SYMBOL	Símbolo de moneda local.
CRNCYSTR	El mismo valor de CURRENCY_SYMBOL.
MON_DECIMAL_POINT	Caracter de punto decimal.
MON_THOUSANDS_SEP	Separador de miles (grupos de tres dígitos).
MON_GROUPING	Como el elemento 'grouping'.
POSITIVE_SIGN	Signo para valores positivos.
NEGATIVE_SIGN	Signo para valores negativos.
INT_FRAC_DIGITS	Dígitos fraccionarios internacionales.
FRAC_DIGITS	Dígitos fraccionarios locales.
P_CS_PRECEDES	Devuelve 1 si CURRENCY_SYMBOL precede un valor positivo.
P_SEP_BY_SPACE	Devuelve 1 si un espacio separa a CURRENCY_SYMBOL de un valor positivo.

Constante	Descripción
N_CS_PRECEDES	Devuelve 1 si CURRENCY_SYMBOL precede un valor negativo.
N_SEP_BY_SPACE	Devuelve 1 si un espacio separa a CURRENCY_SYMBOL de un valor negativo.

Constante	Descripción
P_SIGN_POSN	• Devuelve 0 si unos paréntesis rodean la cantidad y currency_symbol. • Devuelve 1 si la cadena de signo precede a la cantidad y currency_symbol. • Devuelve 2 si la cadena de signo sigue a continuación de la cantidad y currency_symbol. • Devuelve 3 si la cadena de signo precede inmediatamente a currency_symbol. • Devuelve 4 si la cadena de signo sigue inmediatamente después de
N_SIGN_POSN	

Constante	Descripción
Constantes de la Categoría LC_NUMERIC	
DECIMAL_POINT	Caracter de punto decimal.
RADIXCHAR	El mismo valor de DECIMAL_POINT.
THOUSANDS_SEP	Caracter separador para los miles (grupos de tres dígitos).
THOUSEP	El mismo valor de THOUSANDS_SEP.
GROUPING	
Constantes de la Categoría LC_MESSAGES	
YESEXPR	Cadena de expresión regular para coincidir con la entrada 'sí'.
NOEXPR	Cadena de expresión regular para coincidir con la entrada 'no'.
YESSTR	Cadena de salida para 'sí'.
NOSTR	Cadena de salida para 'no'.
Constantes de la Categoría LC_CTYPE	
CODESET	Devuelve una cadena con el nombre de la codificación de caracteres.

Nota: Esta función no está implementada en plataformas Windows.

Vea también **setlocale()** y **localeconv()**.

nl2br

(PHP 3, PHP 4 , PHP 5)

nl2br -- Inserta saltos de linea HTML antes de cada salto de línea

Descripción

string **nl2br** (string cadena)

Devuelve la *cadena* con '<br />' insertados antes de cada nueva línea.

> **Nota:** A partir de PHP 4.0.5, **nl2br()** respeta los estándares de XHTML. Todas las versiones anteriores de PHP a la 4.0.5 devolverán *string* con '<br>' antes de cada nueva línea, en vez de '<br />'.

Ejemplo 1. Aplicando nl2br()

```
<?php
echo nl2br("esto es una\nprueba");
?>
```

Salida:

```
esto es una<br />
prueba
```

Vea también **htmlspecialchars()**, **htmlentities()** y **wordwrap()**.

number_format

(PHP 3, PHP 4 , PHP 5)

number_format -- Dar formato a un número con los miles agrupados

Descripción

string **number_format** (float numero [, int decimales [, string punto_dec, string sep_miles]])

number_format() devuelve una versión con formato de *numero*. Esta función acepta uno, dos, o cuatro parámetros (no tres):

Si solo se entrega un parámetro, *numero* recibirá un formato sin decimales, pero con una coma (",") entre cada grupo de miles.

Si se entregan dos parámetros, *numero* recibirá un formato con la cantidad de *decimales* dada, con un punto (".") al frente, y una coma (",") entre cada grupo de miles.

Si todos los cuatro parámetros son dados, *numero* recibirá un formato con la cantidad de *decimales* dada, *punto_dec* en lugar de un punto (".") antes de los decimales, y *sep_miles* en lugar de una coma (",") entre cada grupo de miles.

Sólo el primer caracter de *sep_miles* es usado. Por ejemplo, si usa *foo* como *sep_miles* sobre el número *1000*, **number_format()** devolverá *1f000*.

Ejemplo 1. Ejemplo de number_format()

Por ejemplo, la notación francesa usa usualmente dos decimales, coma (',') como separador decimal, y espacio (' ') como separador de miles. Esto se puede lograr con esta línea:

```
<?php

$numero = 1234.56;

// notacion inglesa (predeterminada)
$numero_formato_ingles = number_format($numero);
// 1,234

// notacion francesa
$numero_formato_frances = number_format($numero, 2, ',', ' ');
// 1 234,56

$numero = 1234.5678;

// notacion inglesa sin separador de miles
$numero_formato_ingles = number_format($numero, 2, '.', '');
// 1234.57

?>
```

Vea también: **sprintf()**, **printf()** y **sscanf()**.

ord

(PHP 3, PHP 4 , PHP 5)

ord -- Devuelve el valor ASCII de un caracter

Descripción

int **ord** (string cadena)

Devuelve el valor ASCII del primer caracter de *cadena*. Esta función complementa a **chr()**.

Ejemplo 1. Ejemplo de ord()

```
if (ord ($cad) == 10) {
    echo "El primer caracter de \$cad es un salto de línea.\n";
}
```

Vea también **chr()**.

parse_str

(PHP 3, PHP 4 , PHP 5)

parse_str -- Divide la cadena en variables

Descripción

void **parse_str** (string cad)

Divide *cad* como si fuera la cadena de consulta enviada por un URL y crea las variables en el ámbito actual.

Ejemplo 1. Usando parse_str()

```
$cad = "primero=valor&segundo[]=esto+funciona&segundo[]=otro";
parse_str($cad);
echo $primero;    /* escribe "valor" */
echo $segundo[0]; /* escribe "esto funciona" */
echo $segundo[1]; /* escribe "otro" */
```

print

(PHP 3, PHP 4, PHP 5)

print -- Emite una cadena

Descripción

print (string arg)

Emite *arg*.

Vea también: **echo()**, **printf()**, y **flush()**.

printf

(PHP 3, PHP 4 , PHP 5)

printf -- Imprimir una cadena con formato

Descripción

int **printf** (string formato [, mixed args [, mixed ...]])

Produce una salida de acuerdo con el *formato*, el cual se describe en la documentación de **sprintf()**.

Devuelve la longitud de la cadena impresa.

Vea también **print()**, **sprintf()**, **vprintf()**, **sscanf()**, **fscanf()**, y **flush()**.

quoted_printable_decode

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

quoted_printable_decode -- Convierte una cadena con marcación imprimible a una cadena de 8 bits

Descripción

string **quoted_printable_decode** (string cad)

Esta función devuelve una cadena binaria de 8 bit que se corresponde con la cadena con marcación imprimible decodificada. Esta función es similar a **imap_qprint()**, pero sin requerir que el módulo IMAP funcione.

quotemeta

(PHP 3, PHP 4 , PHP 5)

quotemeta -- Quote meta characters

Descripción

string **quotemeta** (string cad)

Devuelve una versión de la cadena con una barra invertida (\) antes de cada caracter de este conjunto:

```
. \\ + * ? [ ^ ] ( $ )
```

Vea también **addslashes()**, **htmlentities()**, **htmlspecialchars()**, **nl2br()**, y **stripslashes()**.

rtrim

(PHP 3, PHP 4 , PHP 5)

rtrim -- Elimina espacios en blanco al final de la cadena.

Descripción

string **rtrim** (string cad)

Devuelve la cadena argumento sin espacios en blanco ni saltos de línea al final. Es un alias para **chop()**.

Ejemplo 1. Ejemplo de rtrim()

```
$recortada = rtrim ($linea);
```

Vea también **trim()**, **ltrim()**.

setlocale

(PHP 3, PHP 4 , PHP 5)

setlocale -- Fija la información de localidad

Descripción

string **setlocale** (string categoria, string localidad)

categoria es una cadena que especifica la categoría de las funciones afectadas por el ajuste de localidad:

- LC_ALL para todas las funciones
- LC_COLLATE para la comparación de cadenas - aún no incluída en el PHP
- LC_CTYPE para la conversión y clasificación de caracteres, como por ejemplo **strtoupper()**

- LC_MONETARY para localeconv() - aún no incluída en el PHP
- LC_NUMERIC para el separador decimal
- LC_TIME para el formato de fecha y hora con **strftime()**

Si *localidad* es la cadena vacía *""*, los nombres de localidad se fijarán a partir de las variables de entorno con los mismos nombres de las categorías anteriores, o desde "LANG".

Si la localidad es cero o *"0"*, el ajuste de localidad no se ve afectado y sólo se devuelve el ajuste actual.

setlocale devuelve la nueva localidad, o **FALSE** si la funcionalidad de localización no está disponible en la plataforma, la localidad especificada no existe o el nombre de categoría no es válido. Un nombre de categoría no válido también produce un mensaje de aviso.

sha1_file

(PHP 4 >= 4.3.0, PHP 5)

sha1_file -- Calcular el resumen criptográfico sha1 de un archivo

Descripción

string **sha1_file** (string nombre_archivo [, bool salida_pura])

Calcula el resumen criptográfico sha1 de *nombre_archivo* usando el Algoritmo de Hash Seguro US 1, y devuelve ese resumen. El resumen criptográfico es un número hexadecimal de 40 caracteres. En caso de fallo, se devuelve **FALSE**. Si el parámetro opcional *salida_pura* es definido a **TRUE**, entonces el resumen sha1 es devuelto en su lugar en un formato binario puro, con una longitud de 20.

> **Nota:** El parámetro opcional *salida_pura* fue añadido en PHP 5.0.0 y tiene un valor predeterminado de **FALSE**.

Vea también **sha1()**, **crc32()**, y **md5_file()**

sha1

(PHP 4 >= 4.3.0, PHP 5)

sha1 -- Calcular el resumen criptográfico sha1 de una cadena

Descripción

string **sha1** (string cadena [, bool salida_pura])

Calcula el resumen criptográfico sha1 de *cadena* usando el Algoritmo de Hash Seguro US 1, y devuelve ese resumen. El resumen criptográfico es un número hexadecimal de 40 caracteres. Si el parámetro opcional *salida_pura* es definido a **TRUE**, entonces el resumen sha1 es devuelto en su lugar en un formato binario puro, con una longitud de 20.

Nota: El parámetro opcional *salida_pura* fue añadido en PHP 5.0.0 y tiene un valor predeterminado de **FALSE**

Ejemplo 1. Un ejemplo de sha1()

```
<?php
$cadena = 'manzana';

if (sha1($cadena) === '6cc76bddf29e934258e30fbf301ad8ec89cb7cc6') {
    echo "&iquest;Quisiera una manzana verde o roja?";
    exit;
}
?>
```

Vea también **sha1_file()**, **crc32()**, y **md5()**

similar_text

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

similar_text -- Calcula la similitud entre dos cadenas

Descripción

int **similar_text** (string primera, string segunda [, double porcentaje])

Esta función calcula la similitud entre dos cadenas según se describe en Oliver [1993]. Nótese que esta implementación no utiliza una pila como en el pseudo-código de Oliver, sino llamadas recursivas que pueden o no acelerar el proceso completo. Nótese también que la complejidad de este algoritmo es O(N**3), donde N es la longitud de la cadena más larga.

Pasando una referencia como tercer argumento, **similar_text()** calculará para usted la similitud como porcentaje. Devuelve el número de caracteres coincidentes en ambas cadenas.

soundex

(PHP 3, PHP 4 , PHP 5)

soundex -- Calcula la clave soundex de una cadena

Descripción

string **soundex** (string cad)

Calcula la clave soundex de *cad*.

Las claves soundex tienen la propiedad de que las palabras que se pronuncian de forma parecida tienen la misma clave, de modo que se pueden usar para simplificar la búsqueda en las bases de datos cuando se conoce la pronunciación pero no la transcripción. Esta función soundex devuelve una cadena de 4 caracteres que comienza por una letra.

Esta función soundex en particular es la descrita por Donald Knuth en "The Art Of Computer Programming, vol. 3: Sorting And Searching", Addison-Wesley (1973), pp. 391-392.

Ejemplo 1. Ejemplos de Soundex

```
soundex ("Euler") == soundex ("Ellery") == 'E460';
soundex ("Gauss") == soundex ("Ghosh") == 'G200';
soundex ("Knuth") == soundex ("Kant") == 'H416';
soundex ("Lloyd") == soundex ("Ladd") == 'L300';
soundex ("Lukasiewicz") == soundex ("Lissajous") == 'L222';
```

sprintf

(PHP 3, PHP 4 , PHP 5)

sprintf -- Devuelve una cadena con formato

Descripción

string **sprintf** (string formato [, mixed args [, mixed ...]])

Devuelve una cadena producida de acuerdo con la cadena de formato *formato*.

La cadena de formato se compone de cero o más directivas: caracteres ordinarios (excluyendo *%*) que son copiados directamente en el resultado, y unas *especificaciones de conversión*, cada una de las cuales produce una búsqueda por su propio parámetro. Esto se aplica tanto en **sprintf()** como en **printf()**.

Cada especificación de conversión consiste de un signo de porcentaje (*%*), seguido por uno o más de los siguientes elementos, en orden:

1. Un *indicador de signo* opcional que obliga a que se use un determinado signo (- o +) en un número. De forma predeterminada, sólo el signo - es usado en un número si éste es negativo. Este indicador obliga a los números positivos a que tengan también el signo + adjunto, comportamiento que se agregó en PHP 4.3.0.

2. Un *indicador de relleno* opcional, que dice qué caracter será usado para adaptar el resultado al tamaño de cadena apropiado. Este puede ser un caracter de espacio, o un *0* (caracter cero). El comportamiento predeterminado es rellenar con espacios. Un caracter de relleno alternativo puede especificarse al colocar una comilla sencilla (*'*) al comienzo. Vea los ejemplos más adelante.

3. Un *indicador de alineamiento* opcional que dice si el resultado debe alinearse a la izquierda o a la derecha. El comportamiento predeterminado es alinear a la derecha; un caracter - en este lugar hace que la alineación sea a la izquierda.

4. Un número opcional, un *indicador de ancho* que dice cuántos caracteres (como mínimo) debe producir la conversión.

5. Un *indicador de precisión* opcional que dice cuántos dígitos decimales deben mostrarse para los números de punto flotante. Cuando se usa este indicador con una cadena, actúa como un punto de corte, indicando un límite máximo de caracteres para la cadena.

6. Un *indicador de tipo* que especifica el tipo bajo el que deben tratarse los datos del argumento. Los posibles tipos son:

   % - un caracter de porcentaje literal. No requiere argumento.

b - el argumento es tratado como un entero, presentado como un número binario.

c - el argumento es tratado como un entero, y presentado como el caracter con ese valor ASCII.

d - el argumento es tratado como un entero, y presentado como un número decimal (con signo).

e - el argumento es tratado como notación científica (p.ej. 1.2e+2).

u - el argumento es tratado como un entero, y presentado como un número decimal sin signo.

f - el argumento es tratado como un flotante, y presentado como un número de punto flotante.

o - el argumento es tratado como un entero, y presentado como un número octal.

s - el argumento es tratado y presentado como una cadena.

x - el argumento es tratado como un entero y presentado como un número hexadecimal (con letras minúsculas).

X - el argumento es tratado como un entero y presentado como un número hexadecimal (con letras mayúsculas).

A partir de PHP 4.0.6, la cadena de formato soporta la numeración/intercambio de argumentos. He aquí un ejemplo:

Ejemplo 1. Intercambio de argumentos

```
<?php
$formato = "Hay %d monos en el %s";
printf($formato, $num, $ubicacion);
?>
```

Esto podría imprimir, "Hay 5 monos en el árbol". Pero imagine que creamos una cadena de formato en un archivo separado, generalmente por que queremos implementar un mecanismo de internacionalización, y re-escribimos el código:

Ejemplo 2. Intercambio de argumentos

```
<?php
$formato = "El %s contiene %d monos";
printf($formato, $num, $ubicacion);
?>
```

Ahora tenemos un problema. El orden de los recipientes en la cadena de formato no coincide con el orden de los argumentos en el código. Quisiéramos dejar el código tal como está, y simplemente indicar en la cadena de formato cuáles argumentos están siendo referidos por los recipientes. Entonces re-escribiríamos la cadena de formato de esta forma:

Ejemplo 3. Intercambio de argumentos

```
<?php
$formato = "El %2\$s contiene %1\$d monos";
printf($formato, $num, $ubicacion);
?>
```

Un beneficio adicional de esto es que puede repetir los recipientes sin agregar más argumentos en el código. Por ejemplo:

Ejemplo 4. Intercambio de argumentos

```
<?php
$formato = "El %2\$s contiene %1\$d monos.
            Se trata de un bonito %2\$s lleno con %1\$d monos.";
printf($formato, $num, $ubicacion);
?>
```

Vea también **printf()**, **sscanf()**, **fscanf()**, **vsprintf()**, y **number_format()**.

Ejemplos

Ejemplo 5. printf(): ejemplos varios

```php
<?php
$n =  43951789;
$u = -43951789;
$c = 65; // el valor ASCII 65 es 'A'

// note el doble %%, esto imprime un caracter '%' literal
printf("%%b = '%b'\n", $n); // representacion binaria
printf("%%c = '%c'\n", $c); // imprime el caracter ascii, igual que la funcion chr()
printf("%%d = '%d'\n", $n); // representacion de entero estandar
printf("%%e = '%e'\n", $n); // notacion cientifica
printf("%%u = '%u'\n", $n); // representacion entera sin signo de un entero positivo
printf("%%u = '%u'\n", $u); // representacion entera sin signo de un entero negativo
printf("%%f = '%f'\n", $n); // representacion en punto flotante
printf("%%o = '%o'\n", $n); // representacion octal
printf("%%s = '%s'\n", $n); // representacion de cadena
printf("%%x = '%x'\n", $n); // representacion hexadecimal (minusculas)
printf("%%X = '%X'\n", $n); // representacion hexadecimal (mayusculas)

printf("%%+d = '%+d'\n", $n); // indicador de signo en un entero positivo
printf("%%+d = '%+d'\n", $u); // indicador de signo en un entero negativo
?>
```

La salida de este programa sería:

```
%b = '10100111101010011010101101'
%c = 'A'
%d = '43951789'
%e = '4.39518e+7'
%u = '43951789'
%u = '4251015507'
%f = '43951789.000000'
%o = '247523255'
%s = '43951789'
%x = '29ea6ad'
%X = '29EA6AD'
%+d = '+43951789'
%+d = '-43951789'
```

Ejemplo 6. printf(): indicadores de cadena

```php
<?php
$s = 'mono';
$t = 'varios monos';

printf("[%s]\n",      $s); // salida de cadena estandar
printf("[%10s]\n",    $s); // alineacion a derecha con espacios
printf("[%-10s]\n",   $s); // alineacion a izquierda con espacios
printf("[%010s]\n",   $s); // el relleno con ceros funciona con cadenas tambien
printf("[%'#10s]\n",  $s); // usar el caracter de relleno '#'
printf("[%10.10s]\n", $t); // alineacion a izquierda pero con un corte de 10 caracteres
?>
```

La salida de este programa sería:

```
[mono]
[      mono]
[mono      ]
[000000mono]
[######mono]
[varios mon]
```

Ejemplo 7. sprintf(): enteros con relleno de ceros

```php
<?php
$fecha_iso = sprintf("%04d-%02d-%02d", $anyo, $mes, $dia);
?>
```

Ejemplo 8. sprintf(): formato de valores monetarios

```
<?php
$dinero1 = 68.75;
$dinero2 = 54.35;
$dinero = $dinero1 + $dinero2;
// echo $dinero imprimiria "123.1";
$con_formato = sprintf("%01.2f", $dinero);
// echo $con_formato imprime "123.10"
?>
```

Ejemplo 9. sprintf(): notación científica

```
<?php
$numero = 362525200;

echo sprintf("%.3e", $numero); // imprime 3.63e+8
?>
```

sscanf

(PHP 4 >= 4.0.1, PHP 5)

sscanf -- Trocea la entrada desde una cadena según un formato dado

Descripción

mixed **sscanf** (string cad, string formato [, string var1])

La función**sscanf()** es la función de entrada análoga de **printf()**. **sscanf()** lee del parámetro de cadena *cad* y lo interpreta según el *formato* especificado. Si sólo se pasan dos parámetros a esta función, los valores devueltos se harán en una matriz.

Ejemplo 1. Ejemplo de sscanf()

```
// obteniendo el número de serie
$numserie = sscanf("SN/2350001","SN/%d");
// y la fecha de fabricación
$fecha = "01 Enero 2000";
list($dia, $mes, $anno) = sscanf($fecha,"%d %s %d");
echo "El objeto $numserie fue fabricado el: $anno-".substr($mes,0,3)."-$dia\n";
```

Si se pasan los parámetros opcionales, la función devolverá el número de valores asignados. Los parámetros opcionales deben ser pasados por referencia.

Ejemplo 2. Ejemplo de sscanf() - usando parámetros opcionales

```
// obtener autor y generar la ficha DocBook
$autor = "24\tLewis Carroll";
$n = sscanf($autor,"%d\t%s %s", &$id, &$nombre, &$apell);
echo "<autor id='$id'>
        <firstname>$nombre</firstname>
        <surname>$apell</surname>
</author>\n";
```

Vea también: **fscanf()**, **printf()**, y **sprintf()**.

str_ireplace

(PHP 5)

str_ireplace -- Versión insensible a mayúsculas y minúsculas de **str_replace()**.

Descripción

mixed **str_ireplace** (mixed busqueda, mixed reemplazo, mixed asunto [, int &conteo])

Esta función devuelve una cadena o una matriz con todas las ocurrencias de *busqueda* en *asunto* (ignorando mayúsculas y minúsculas) sustituidas por el *reemplazo* dado. Si no necesita reglas de reemplazo sofisticadas, usted debería, por lo general, usar esta función en lugar de **eregi_replace()** o **preg_replace()** con el modificador *i*.

Si *asunto* es una matriz, entonces las búsquedas y reemplazos son realizados con cada entrada de *asunto*, y el valor devuelto es asimismo una matriz.

Si *busqueda* y *reemplazo* son matrices, entonces **str_ireplace()** toma un valor de cada matriz y los usa para realizar las búsquedas y reemplazos sobre *asunto*. Si *reemplazo* tiene menos valores que *busqueda*, entonces se usa una cadena vacía para el resto de valores de reemplazo. Si *busqueda* es una matriz y *reemplazo* es una cadena, entonces ésta cadena de reemplazo es usada para cada valor de *busqueda*.

Ejemplo 1. Ejemplo de str_ireplace()

```
<?php
$etiqueta_body = str_ireplace("%body%", "black", "<body text=%BODY%>");
?>
```

Esta función es segura con material binario.

> **Nota:** A partir de PHP 5.0.0, el número de *agujas* encontradas y reemplazadas será devuelto en *conteo*, el cual es pasado por referencia. Antes de PHP 5.0.0, este parámetro no se encuentra disponible.

Vea también: **str_replace()**, **ereg_replace()**, **preg_replace()**, y **strtr()**.

str_pad

(PHP 4 >= 4.0.1, PHP 5)

str_pad -- Rellena una cadena con otra hasta una longitud dada

Descripción

string **str_pad** (string entrada, int tama_relleno [, string cad_relleno [, int tipo_relleno]])

Esta función rellena la cadena *entrada* por la derecha, la izquierda o por ambos lados hasta el largo indicado. Si no se especifica el argumento opcional *cad_relleno*, *entrada* es rellenada con espacios. En caso contrario, será rellenada con los caracteres de *cad_relleno* hasta el límite.

El argumento opcional *tipo_relleno* puede valer STR_PAD_RIGHT, STR_PAD_LEFT, o STR_PAD_BOTH. Si no se especifica, se asume que vale STR_PAD_RIGHT.

Si el valor de *tama_relleno* es negativo o menor que la longitud de la cadena de entrada, no se produce rellenado alguno.

Ejemplo 1. Ejemplo de str_pad()

```
$entrada = "Alien";
print str_pad($entrada, 10);                      // produce "Alien     "
print str_pad($entrada, 10, "-=", STR_PAD_LEFT);  // produce "-=-=-Alien"
print str_pad($entrada, 10, "_", STR_PAD_BOTH);   // produce "__Alien___"
```

str_repeat

(PHP 4 , PHP 5)

str_repeat -- Repite una cadena

Descripción

string **str_repeat** (string cad_entrada, int veces)

Devuelve la *cad_entrada* repetida *veces*. *veces* debe ser mayor que 0.

Ejemplo 1. Ejemplo de str_repeat()

```
echo str_repeat ("-=", 10);
```

Esto mostrará "-=-=-=-=-=-=-=-=-=-=".

> **Nota:** Esta función fue añadida en el PHP 4.0.

str_replace

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

str_replace -- Sustitiye todas las apariciones de la aguja en el pajar por la cadena

Descripción

string **str_replace** (string aguja, string cad, string pajar)

Esta función sustituye todas las apariciones de la *aguja* en el *pajar* por la *cad* dada. Si no precisa reglas especiales de sustitución, deberá usar siempre esta función en lugar de **ereg_replace()**.

Ejemplo 1. Ejemplo de str_replace()

```
$bodytag = str_replace ("%cuerpo%", "negro", "<body text=%cuerpo%>");
```

Esta función tiene seguridad binaria.

> **Nota: str_replace()** fue añadida en PHP 3.0.6, pero tuvo errores hasta el PHP 3.0.8.

Vea también **ereg_replace()** y **strtr()**.

str_rot13

(PHP 4 >= 4.2.0, PHP 5)

str_rot13 -- Realizar la transformación rot13 sobre una cadena

Descripción

string **str_rot13** (string cadena)

Esta función aplica la codificación ROT13 sobre el argumento *cadena* y devuelve la cadena resultante. La codificación ROT13 simplemente desplaza cada letra en 13 posiciones en el alfabeto, mientras que deja todos los caracteres no-alfabéticos intactos. La codificación y la decodificación se realizan con la misma función, pasar una cadena codificada como argumento devolverá la versión original.

Ejemplo 1. Ejemplo de str_rot13()

```
<?php

echo str_rot13('PHP 4.3.0'); // CUC 4.3.0

?>
```

> **Nota:** El comportamiento de esta función era ñocado hasta PHP 4.3.0. Anteriormente, la *cadena* era también modificada, como si fuera pasada por referencia.

str_shuffle

(PHP 4 >= 4.3.0, PHP 5)

str_shuffle -- Reordena aleatoriamente una cadena

Descripción

string **str_shuffle** (string cadena)

str_shuffle() baraja una cadena. Es creada una permutación de todas las posibles.

Ejemplo 1. Ejemplo de str_shuffle()

```
<?php
$cadena = 'abcdef';
$desordenada = str_shuffle($cadena);

// Esto imprime algo como: bfdaec
echo $desordenada;
?>
```

Vea también **shuffle()** y **rand()**.

str_split

(PHP 5)

str_split -- Convertir una cadena en una matriz

Descripción

array **str_split** (string cadena [, int longitud_separacion])

Convierte una cadena en una matriz. Si el parámetro opcional *longitud_separacion* es especificado,

la matriz devuelta estará separada en pedazos, cada uno de longitud *longitud_separacion*, de otro modo cada trozo tendrá un caracter de longitud.

Se devuelve **FALSE** si *longitud_separacion* es menor que 1. Si la longitud *longitud_separacion* excede la longitud de *cadena*, entonces la cadena completa es devuelta como el primer (y único) elemento de la matriz.

Ejemplo 1. Ejemplos de uso de str_split()

```
<?php

$cadena = "Hello Friend";

$matriz1 = str_split($cadena);
$matriz2 = str_split($cadena, 3);

print_r($matriz1);
print_r($matriz2);


?>
```

La salida puede verse algo como:

```
Array
(
    [0] => H
    [1] => e
    [2] => l
    [3] => l
    [4] => o
    [5] =>
    [6] => F
    [7] => r
    [8] => i
    [9] => e
    [10] => n
    [11] => d
)

Array
(
    [0] => Hel
    [1] => lo
    [2] => Fri
    [3] => end
)
```

Ejemplo 2. Ejemplos relacionados con str_split()

```
<?php

$cadena = "Hello Friend";

echo $cadena{0};  // H
echo $cadena{8};  // i

// Crea: array('H','e','l','l','o',' ','F','r','i','e','n','d')
$matriz1 = preg_split('//', $cadena, -1, PREG_SPLIT_NO_EMPTY);

?>
```

Vea también **chunk_split()**, **preg_split()**, **split()**, **count_chars()**, **str_word_count()**, y for.

str_word_count

(PHP 4 >= 4.3.0, PHP 5)

str_word_count -- Devolver información sobre las palabras usadas en una cadena

Descripción

mixed **str_word_count** (string cadena [, int formato])

Cuenta el número de palabras en *cadena*. Si el *formato* opcional no es especificado, entonces el valor devuelto será un número que representa el número de palabras encontradas. En caso de que el *formato* sea especificado, el valor devuelto será una matriz, cuyo contenido depende del *formato*. Los valores posibles para el *formato* y las salidas resultantes son listadas a continuación.

- 1 - devuelve una matriz que contiene todas las palabras encontradas en la *cadena*.
- 2 - devuelve una matriz asociativa, en donde la clave es la posición numérica de la palabra dentro de la *cadena*, y el valor es la palabra misma.

Para el propósito de esta función, 'palabra' se define como una cadena dependiente de la localidad, que contiene caracteres alfabéticos, la cual puede también contener, pero no comenzar por los caracteres "'" y "-".

Ejemplo 1. Ejemplo de uso de str_word_count()

```
<?php

$cadena = "Hola amigo, se ve bien
           el dia de hoy.";

$a   = str_word_count($cadena, 1);
$b   = str_word_count($cadena, 2);
$c   = str_word_count($cadena);

print_r($a);
print_r($b);
echo $c;
?>
```

La salida puede verse algo asi:

```
Array
(
    [0] => Hola
    [1] => amigo
    [2] => se
    [3] => ve
    [4] => bien
    [5] => el
    [6] => dia
    [7] => de
    [8] => hoy
)

Array
(
    [0] => Hola
    [5] => amigo
    [12] => se
    [15] => ve
    [18] => bien
    [34] => el
    [37] => dia
    [41] => de
    [44] => hoy
)

9
```

Vea también **explode()**, **preg_split()**, **split()**, **count_chars()**, y **substr_count()**.

strcasecmp

(PHP 3>= 3.0.2, PHP 4 , PHP 5)

strcasecmp -- Comparación de cadenas insensible a mayúsculas y minúsculas y segura en modo binario

Descripción

int **strcasecmp** (string cad1, string cad2)

Devuelve < 0 si *cad1* es menor que *cad2*; > 0 si *cad1* es mayor que *cad2*, y 0 si son iguales.

Ejemplo 1. Ejemplo de strcasecmp()

```
$var1 = "Hello";
$var2 = "hello";
if (!strcasecmp ($var1, $var2)) {
    echo '$var1 es igual a $var2 en una comparación sin tener en cuenta '
     .'mayúsculas o minúsculas';
}
```

Vea también **ereg()**, **strcmp()**, **substr()**, **stristr()**, y **strstr()**.

strchr

(PHP 3, PHP 4 , PHP 5)

strchr -- Encuentra la primera aparición de un caracter

Descripción

string **strchr** (string pajar, string aguja)

Esta función es un alias para **strstr()**, y es idéntica en todo.

strcmp

(PHP 3, PHP 4 , PHP 5)

strcmp -- Comparación de cadenas con seguridad binaria

Descripción

int **strcmp** (string cad1, string cad2)

Devuelve < 0 si *cad1* es menor que *cad2*; > 0 si *cad1* es mayor que *cad2*, y 0 si son iguales.

Nótese que esta comparación es sensible a mayúsculas y minúsculas.

Vea también **ereg()**, **strcasecmp()**, **substr()**, **stristr()**, **strncmp()**, y **strstr()**.

strcoll

(PHP 4 >= 4.0.5, PHP 5)

strcoll -- Comparación de cadenas basada en la localidad

Descripción

int **strcoll** (string cad1, string cad2)

Devuelve < 0 si *cad1* es menor que *cad2*; > 0 si *cad1* es mayor que *cad2*, y 0 si son ñalentes. **strcoll ()** usa la localidad actual para realizar las comparaciones. Si la localidad actual es C o POSIX, esta función es ñalente a **strcmp()**.

Note que esta comparación es sensible a mayúsculas y minúsculas, y a diferencia de **strcmp()**, esta función no es segura con material binario.

> **Nota: strcoll()** fue agregada en PHP 4.0.5, pero no estuvo habilitada para win32 hasta 4.2.3.

Vea también **ereg()**, **strcmp()**, **strcasecmp()**, **substr()**, **stristr()**, **strncasecmp()**, **strncmp()**, **strstr ()**, y **setlocale()**.

strcspn

(PHP 3>= 3.0.3, PHP 4 , PHP 5)

strcspn -- Encuentra la longitud del elemento inicial que no coincide con la máscara

Descripción

int **strcspn** (string cad1, string cad2)

Devuelve la longitud del segmento inicial de *cad1* que *no* contiene ninguno de los caracteres de *cad2*.

Vea también **strspn()**.

strip_tags

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

strip_tags -- Elimina marcas HTML y PHP de una cadena

Descripción

string **strip_tags** (string cad [, string etiq_permitidas])

Esta función intenta eliminar todas las etiquetas HTML y PHP de la cadena dada. Causa error por

precaución en caso de etiquetas incompletas o falsas. Utiliza la misma máquina de estados para eliminar las etiquetas que la función **fgetss()**.

Puede usar el parámetro opcional para especificar las etiquetas que no deben eliminarse.

> **Nota:** *etiq_permitidas* fue añadido en PHP 3.0.13, PHP4B3.

stripcslashes

(PHP 4 , PHP 5)

stripcslashes -- Desmarca la cadena marcada con **addcslashes()**

Descripción

string **stripcslashes** (string cad)

Devuelve una cadena con las barras invertidas eliminadas. Reconoce las marcas tipo C *\n*, *\r* ..., y la representación octal y hexadecimal.

> **Nota:** Añadida en PHP4b3-dev.

Vea también **addcslashes()**.

stripos

(PHP 5)

stripos -- Encontrar la posición de la primera ocurrencia de una cadena, insensible a mayúsculas y minúsculas

Descripción

int **stripos** (string pajar, string aguja [, int desplazamiento])

Devuelve la posición numérica de la primera ocurrencia de *aguja* en el *pajar* tipo **string**. A diferencia de **strpos()**, **stripos()** es indiferente a mayúsculas y minúsculas.

Note que la *aguja* puede ser una cadena de uno o más caracteres.

Si *aguja* no se encuentra, **stripos()** devolverá el valor **boolean** **`FALSE`**.

Aviso
Esta función puede devolver **`FALSE`**, pero también puede devolver un valor no-booleano que será evaluado **`FALSE`**, como por ejemplo *0* o "". Por favor, lea la sección Booleans para más información. Utilice el operador === para comprobar el valor devuelto por esta función.

Ejemplo 1. Ejemplos de stripos()

```php
<?php
$encontradme = 'a';
$micadena1 = 'xyz';
$micadena2 = 'ABC';

$pos1 = stripos($micadena1, $encontradme);
$pos2 = stripos($micadena2, $encontradme);

// No, ciertamente 'a' no esta en 'xyz'
if ($pos1 === false) {
    echo "La cadena '$encontradme' no fue encontrada en la cadena '$micadena1'";
}

// Note nuestro uso de ===.  Simplemente == no funcionaria como es de
// esperarse, ya que la posicion de 'a' es el caracter 0 (el primero).
if ($pos2 !== false) {
    echo "Encontramos '$encontradme' en '$micadena2' en la posicion $pos2";
}
?>
```

Si *aguja* no es una cadena, es convertida a un entero y aplicada como el valor ordinal de un caracter.

El parámetro opcional *desplazamiento* le permite especificar el caracter en *pajar* a partir del que desea empezar a buscar. La posición devuelta sigue siendo relativa al comienzo de *pajar*.

> **Nota:** Esta función es segura binariamente.

Vea también **strpos()**, **strrpos()**, **strrchr()**, **substr()**, **stristr()**, **strstr()**, **strripos()** y **str_ireplace()**.

stripslashes

(PHP 3, PHP 4 , PHP 5)

stripslashes -- Desmarca la cadena marcada con **addslashes()**

Descripción

string **stripslashes** (string cad)

Devuelve una cadena con las barras invertidas eliminadas (\' se convierte en ', etc.). Las barras invertidas dobles se convierten en sencillas.

Vea también **addslashes()**.

stristr

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

stristr -- **strstr()** sin tener en cuenta mayúsculas o minúsculas

Descripción

string **stristr** (string pajar, string aguja)

Devuelve todo el *pajar* desde la primera aparición de la *aguja*, siendo el *pajar* examinado sin tener en cuenta mayúsculas o minúsculas.

Si la *aguja* no se encuentra, devuelve **`FALSE`**.

Si la *aguja* no es una cadena, es convertida a entero y usada como código de un carácter ASCII.

Vea también **strchr()**, **strrchr()**, **substr()**, y **ereg()**.

strlen

(PHP 3, PHP 4 , PHP 5)

strlen -- Obtiene la longitud de la cadena

Descripción

int **strlen** (string cad)

Devuelve la longitud de la *cadena*.

strnatcasecmp

(PHP 4 , PHP 5)

strnatcasecmp -- Comparación de cadenas insensible a mayúsculas y minúsculas usando un algoritmo de "orden natural"

Descripción

int **strnatcasecmp** (string cad1, string cad2)

Esta función implementa un algoritmo de comparación que ordena las cadenas alfanuméricas como lo haría un ser humano. El comportamiento de esta función es similar a **strnatcmp()**, pero la comparación no es sensible a mayúsculas y minúsculas. Para más información, vea la página de Martin Pool sobre Comparación de Cadenas en Orden Natural.

De forma similar a otras funciones de comparación de cadenas, esta devuelve < 0 si *cad1* es menor que *cad2*; > 0 si *cad1* es mayor que *cad2*, y 0 si son iguales.

Vea también **ereg()**, **strcasecmp()**, **substr()**, **stristr()**, **strcmp()**, **strncmp()**, **strnatcmp()**, y **strstr()**.

strnatcmp

(PHP 4 , PHP 5)

strnatcmp -- Compara cadenas usando un algoritmo de "orden natural"

Descripción

int **strnatcmp** (string cad1, string cad2)

Esta función implementa un algoritmo de comparación que ordena las cadenas alfanuméricas como lo haría un ser humano, que es lo que se denomina "orden natural". A continuación se puede ver un ejemplo de la diferencia entre este algoritmo y los algoritmos de ordenación de cadenas habituales en los ordenadores (utilizados en **strcmp()**):

```
$matriz1 = $matriz2 = array ("img12.png","img10.png","img2.png","img1.png");
echo "Comparación de cadenas estándar\n";
usort($matriz1,"strcmp");
print_r($matriz1);
echo "\nComparación de cadenas en orden natural\n";
usort($matriz2,"strnatcmp");
print_r($matriz2);
```

El código anterior generará la siguiente salida:

```
Comparación de cadenas estándar
Array
(
    [0] => img1.png
    [1] => img10.png
    [2] => img12.png
    [3] => img2.png
)

Comparación de cadenas en orden natural
Array
(
    [0] => img1.png
    [1] => img2.png
    [2] => img10.png
    [3] => img12.png
)
```

Para más información, vea la página de Martin Pool sobre Comparación de Cadenas en Orden Natural.

De forma similar a otras funciones de comparación de cadenas, esta devuelve < 0 si *cad1* es menor que *cad2*; > 0 si *cad1* es mayor que *cad2*, y 0 si son iguales.

Nótese que esta comparación es sensible a mayúsculas y minúsculas.

Vea también **ereg()**, **strcasecmp()**, **substr()**, **stristr()**, **strcmp()**, **strncmp()**, **strnatcasecmp()**, y **strstr()**.

strncasecmp

(PHP 4 >= 4.0.2, PHP 5)

strncasecmp -- Comparación de los primeros n caracteres de cadenas, segura con material binario e insensible a mayúsculas y minúsculas

Descripción

int **strncasecmp** (string cad1, string cad2, int longitud)

Esta función es similar a **strcasecmp()**, con la diferencia de que puede especificar el (límite superior de) número de caracteres (*longitud*) de cada cadena a ser usado en la comparación.

Devuelve < 0 si *cad1* es menor que *cad2*; > 0 si *cad1* es mayor que *cad2*, y 0 si son ñalentes.

Vea también **ereg()**, **strcasecmp()**, **strcmp()**, **substr()**, **stristr()**, y **strstr()**.

strncmp

(PHP 4 , PHP 5)

strncmp -- Comparación de los n primeros caracteres de cadenas, con seguridad binaria

Descripción

int **strncmp** (string cad1, string cad2, int largo)

Esta función es similar a **strcmp()**, con la diferencia que se puede especificar el (límite superior del) número de caractares (*largo*) de cada cadena que se usarán en la comparación. Si alguna de las cadenas es menor que el *largo*, se usará su longitud para la comparación.

Devuelve < 0 si *cad1* es menor que *cad2*; > 0 si *cad1* es mayor que *cad2*, y 0 si son iguales.

Nótese que esta comparación es sensible a mayúsculas y minúsculas.

Vea también **ereg()**, **strcasecmp()**, **substr()**, **stristr()**, **strcmp()**, y **strstr()**.

strpbrk

(PHP 5)

strpbrk -- Busca una cadena por cualquiera de los elementos de un conjunto de caracteres

Descripción

string **strpbrk** (string pajar, string lista_caracteres)

strpbrk() busca la cadena *pajar* por una *lista_caracteres*, y devuelve una cadena que empieza desde el caracter encontrado (o **FALSE** si no se encuentra).

> **Nota:** El parámetro *lista_caracteres* es sensible a mayúsculas y minúsculas.

Ejemplo 1. Ejemplo de strpbrk()

```
<?php

$texto = 'Este es un texto Simple.';

// esto imprime "e es un texto Simple." ya que 'e' coincide primero
echo strpbrk($texto, 'me');

// esto imprime "Simple." ya que los caracteres son sensibles a mayusculas/minusculas
echo strpbrk($texto, 'S');
?>
```

strpos

(PHP 3, PHP 4 , PHP 5)

strpos -- Encuentra la posición de la primera aparición de una cadena

Descripción

int **strpos** (string pajar, string aguja [, int desplazamiento])

Devuelve la posición numérica de la primera aparición de la *aguja* en la cadena *pajar*. A diferencia de **strrpos()**, esta función puede tomar una cadena completa como *aguja* y se utilizará en su totalidad.

Si la *aguja* no es hayada, devuelve **FALSE**.

> **Nota:** Es fácil confundir los valores de retorno para "caracter encontrado en la posición 0" y "caracter no encontrado". Aquí se indica cómo detectar la diferencia:

```
// en PHP 4.0b3 y posteriores:
$pos = strpos ($micadena, "b");
if ($pos === false) { // nota: tres signos igual
    // no encontrado ...
}

// en versiones anteriores a la 4.0b3:
$pos = strpos ($micadena, "b");
if (is_string ($pos) && !$pos) {
    // no encontrado ...
}
```

Si la *aguja* no es una cadena, se convierte a entero y se aplica como el valor ordinal de un caracter.

El parámetro opcional *desplazamiento* le permite especificar a partir de qué caracter del *pajar* comenzar a buscar. La posición devuelta es aún relativa al comienzo de *pajar*.

Vea también **strrpos()**, **strrchr()**, **substr()**, **stristr()**, y **strstr()**.

strrchr

(PHP 3, PHP 4 , PHP 5)

strrchr -- Encuentra la última aparición de un caracter en una cadena

Descripción

string **strrchr** (string pajar, string aguja)

Esta función devuelve la porción del *pajar* que comienza en la última aparición de la *aguja* y continúa hasta el final del *pajar*.

Devuelve **FALSE** si la *aguja* no es hallada.

Si la *aguja* contiene más de un caracter, sólo se usará el primero.

Si la *aguja* no es una cadena, se convierte a entero y se aplica como el valor ordinal de un caracter.
Ejemplo 1. Ejemplo de strrchr()

```
// obtener el último directorio de $PATH
$dir = substr (strrchr ($PATH, ":"), 1);

// obtener todo tras el último salto de línea
$texto = "Line 1\nLine 2\nLine 3";
$apell = substr (strrchr ($texto, 10), 1 );
```

Vea también **substr()**, **stristr()**, y **strstr()**.

strrev

(PHP 3, PHP 4 , PHP 5)

strrev -- Invierte una cadena

Descripción

string **strrev** (string cadena)

Devuele la *cadena* invertida.

strripos

(PHP 5)

strripos -- Encontrar la posición de la última ocurrencia de una cadena en otra, insensible a mayúsculas y minúsculas

Descripción

int **strripos** (string pajar, string aguja [, int desplazamiento])

Devuelve la posición numérica de la última ocurrencia de *aguja* en la cadena *pajar*. A diferencia de **strrpos()**, **strripos()** es indiferente a mayúsculas y minúsculas. También note que las posiciones de una cadena comienzan por 0, y no 1.

Note que la *aguja* puede ser una cadena de uno o más caracteres.

Si la *aguja* no es encontrada, se devuelve **FALSE**.

Aviso
Esta función puede devolver **FALSE**, pero también puede devolver un valor no-booleano que será evaluado **FALSE**, como por ejemplo *0* o "". Por favor, lea la sección Booleans para más información. Utilice el operador === para comprobar el valor devuelto por esta función.

Ejemplo 1. Un ejemplo simple de strripos()

```
<?php
$pajar = 'ababcd';
$aguja = 'aB';

$pos   = strripos($pajar, $aguja);

if ($pos === false) {
    echo "Lo siento, no encontramos ($aguja) en ($pajar)";
} else {
    echo "&iexcl;Felicitaciones!\n";
    echo "Encontramos la &uacute;ltima ocurrencia de ($aguja) en ($pajar) en " .
         "la posici&oacute;n ($pos)";
}
?>
```

Produce la salida:

```
&iexcl;Felicitaciones!
Encontramos la &uacute;ltima ocurrencia de (aB) en (ababcd) en la posici&oacute;n (2)
```

El *desplazamiento* puede ser especificado para comenzar la búsqueda en un número arbitrario de caracteres dentro de la cadena. Los valores negativos detendrán la búsqueda en un punto arbitrario anterior al final de la cadena.

Vea también **strrpos()**, **strrchr()**, **substr()**, **stripos()** y **stristr()**.

strrpos

(PHP 3, PHP 4 , PHP 5)

strrpos -- Encuentra la posición de la última aparición de un caracter en una cadena

Descripción

int **strrpos** (string pajar, char aguja)

Devuele la posición numérica de la última aparición de la *aguja* en el *pajar*. Nótese que la aguja en este caso sólo puede ser un caracter único. Si se pasa una cadena como aguja, sólo se utilizará el primer caracter de la misma.

Si la *aguja* no es hayada, devuelve **FALSE**.

Si la *aguja* no es una cadena, se convierte a entero y se aplica como el valor ordinal de un caracter.

Vea también **strpos()**, **strrchr()**, **substr()**, **stristr()**, y **strstr()**.

strspn

(PHP 3>= 3.0.3, PHP 4 , PHP 5)

strspn -- Encuentra la longitud del segmento inicial que coincide con la máscara

Descripción

int **strspn** (string cad1, string cad2)

Devuelve la longitud del segmento inicial de *cad1* que consiste por entero en caracteres contenidos

en *cad2*.

```
strspn ("42 es la respuesta. ¿Cuál es la pregunta ...?", "1234567890");
```

devolverá 2 como resultado.

Vea también **strcspn()**.

strstr

(PHP 3, PHP 4 , PHP 5)

strstr -- Encuentra la primera aparición de una cadena

Descripción

string **strstr** (string pajar, string aguja)

Devuelve todo el *pajar* desde la primera aparición de la *aguja* hasta el final.

Si la *aguja* no es hayada, devuelve **FALSE**.

Si la *aguja* no es una cadena, se convierte a entero y se aplica como el valor ordinal de un caracter.

> **Nota:** Nótese que esta función es sensible a mayúsculas y minúsculas. Para búsquedas no sensibles, utilice **stristr()**.

Ejemplo 1. Ejemplo de strstr()

```
$email = 'sterling@designmultimedia.com';
$dominio = strstr ($email, '@');
print $dominio; // imprime @designmultimedia.com
```

Vea también **stristr()**, **strrchr()**, **substr()**, y **ereg()**.

strtok

(PHP 3, PHP 4 , PHP 5)

strtok -- Divide una cadena en elementos

Descripción

string **strtok** (string arg1, string arg2)

strtok() se usa para dividir en elementos una cadena. Es decir, que si tiene una cadena como "Esta es una cadena de ejemplo" podría dividirla en palabras individuales utilizando el espacio como divisor.

Ejemplo 1. Ejemplo de strtok()

```
$cadena = "Esta es una cadena de ejemplo";
$tok = strtok ($cadena," ");
while ($tok) {
    echo "Palabra=$tok<br>";
    $tok = strtok (" ");
}
```

Nótese que sólo la primera llamada a strtok utiliza el argumento cadena. Cada llamada subsiguiente necesita sólo el divisor a utilizar, puesto que ella guarda la posición actual en la cadena. Para comenzar de nuevo o para dividir otra cadena, simplemente llame a strtok con el argumento de cadena y se inicializará. Nótese que puede poner divisores múltiples como parámetro. La cadena será dividida cuando alguno de los caracteres del argumento sea hallado.

Además tenga cuidado si sus divisores valen "0", pues evalúa como **FALSE** en las expresiones condicionales.

Vea también **split()** y **explode()**.

strtolower

(PHP 3, PHP 4 , PHP 5)

strtolower -- Pasa a minúsculas una cadena

Descripción

string **strtolower** (string cad)

Devuelve la *cadena* con todas sus letras en minúsculas.

Nótese que las letras son definidad por el locale actual. Esto quiere decir que, por ejemplo, en el locale por defecto ("C"), los caracteres como la Ã*ƒ*â€˜ no serán convertidos.

Ejemplo 1. Ejemplo de strtolower()

```
$cad = "María Tenía Un Corderito al que QUERÍA Mucho";
$cad = strtolower($cad);
print $cad; # Visualiza maría tenía un corderito al que quería mucho
```

Vea también **strtoupper()** y **ucfirst()**.

strtoupper

(PHP 3, PHP 4 , PHP 5)

strtoupper -- Pasa a mayúsculas una cadena

Descripción

string **strtoupper** (string cadena)

Devuelve la *cadena* con todas sus letras en mayúsculas.

Nótese que las letras son definidad por el locale actual. Esto quiere decir que, por ejemplo, en el locale por defecto ("C"), los caracteres como la ñ no serán convertidos.

Ejemplo 1. Ejemplo de strtoupper()

```
$cad = "María Tenía Un Corderito al que QUERÍA Mucho";
$cad = strtoupper ($cad);
print $cad; # Visuaiza MARíA TENíA UN CORDERITO AL QUE QUERÍA MUCHO
```

Vea también **strtolower()** and **ucfirst()**.

strtr

(PHP 3, PHP 4 , PHP 5)

strtr -- Traduce ciertos caracteres

Descripción

string **strtr** (string cad, string desde, string hasta)

Esta función trabaja sobre *cad*, traduciendo todas las apariciones de cada caracter en *desde* por el caracter correspondiente en *hasta* y devolviendo el resultado.

Si *desde* y *hasta* son de distinta longitud, los caracteres extra en la más larga son ignorados.

Ejemplo 1. Ejemplo de strtr()

```
$addr = strtr($addr, "ÃƒÂ¤ÃƒÂ¥ÃƒÂ¶", "aao");
```

strtr() puede llamarso sólo con dos argumentos. Si se llama de esta manera, se comporta de otro modo: *desde* debe ser entonces una matriz que contenga pares cadena -> cadena que serán sustituídos en la cadena fuente. **strtr()** siempre buscará la coincidencia más larga primero y *NO* intentará sustituir nada en lo que haya trabajado ya.

Ejemplos:

```
$trad = array ("hola" => "hey", "hey" => "hola");
echo strtr("hey a todos, dije hola", $trad) . "\n";
```

Mostrará: "hola a todos, dije hey",

> **Nota:** Esta característica (2 argumentos) fue añadida en el PHP 4.0

Vea también **ereg_replace()**.

substr_compare

(PHP 5)

substr_compare -- Comparación de 2 cadenas, segura con material binario, opcionalmente insensible a mayúsculas y minúsculas, a partir de un desplazamiento, y hasta un número límite de caracteres

Descripción

int **substr_compare** (string cadena_principal, string cadena, int desplazamiento [, int longitud [, bool insensibilidad_mayusculas]])

substr_compare() compara *cadena_principal* desde la posición *desplazamiento* con *cadena* hasta tantos caracteres como el valor de *longitud*.

Devuelve < 0 si *cadena_principal* desde la posición *desplazamiento* es menor que *cadena*, > 0 si es mayor que *cadena*, y 0 si son iguales. Si *longitud* es igual o mayor que la longitud de *cadena_principal* y *longitud* se define, **substr_compare()** imprime una advertencia y devuelve **FALSE**.

Si *insensibilidad_mayusculas* es **TRUE**, la comparación no distingue entre mayúsculas y minúsculas.

Ejemplo 1. Un ejemplo de substr_compare()

```
<?php
echo substr_compare("abcde", "bc", 1, 2); // 0
echo substr_compare("abcde", "bcg", 1, 2); // 0
echo substr_compare("abcde", "BC", 1, 2, true); // 0
echo substr_compare("abcde", "bc", 1, 3); // 1
echo substr_compare("abcde", "cd", 1, 2); // -1
echo substr_compare("abcde", "abc", 5, 1); // advertencia
?>
```

substr_count

(PHP 4 , PHP 5)

substr_count -- Cuenta el número de apariciones de la subcadena

Descripción

int **substr_count** (string pajar, string aguja)

substr_count() devuelve el número de veces que la subcadena *aguja* se encuentra en la cadena *pajar*.

Ejemplo 1. Ejemplo de substr_count()

```
print substr_count("This is a test", "is"); // prints out 2
```

substr_replace

(PHP 4 , PHP 5)

substr_replace -- Sustituye texto en una parte de una cadena

Descripción

string **substr_replace** (string cadena, string sustituto, int comienzo [, int largo])

substr_replace() sustituye la parte de *cadena* delimitada por los parámetros *comienzo* y (opcionalmente) *largo* por la cadena dada en *sustituto*. Se devuelve el resultado.

Si *comienzo* es positivo, la sustitución comenzará en dicha posición dentro de la *cadena*.

Si *comienzo* es negativo, la sustitución comenzará en dicha posición pero contando desde el final de *cadena*.

Si se especifica el *largo* y es positivo, representa el largo de la porción de *cadena* a sustituir. Si es negativo, representa el número de caracteres desde el final de *cadena* en los que dejar de sustituir. Si no se especifica, valdrá por defecto strlen(*cadena*); es decir, que acabará la sustitución al final de *cadena*.

Ejemplo 1. Ejemplo de substr_replace()

```
<?php
$var = 'ABCDEFGH:/MNRPQR/';
echo "Original: $var<hr>\n";

/* Estos dos ejemplos sustituyen toda $var por 'bob'. */
echo substr_replace ($var, 'bob', 0) . "<br>\n";
echo substr_replace ($var, 'bob', 0, strlen ($var)) . "<br>\n";

/* Inserta 'bob' justo al inicio de $var. */
echo substr_replace ($var, 'bob', 0, 0) . "<br>\n";

/* Los dos siguientes cambian 'MNRPQR' en $var por 'bob'. */
echo substr_replace ($var, 'bob', 10, -1) . "<br>\n";
echo substr_replace ($var, 'bob', -7, -1) . "<br>\n";

/* Borrar 'MNRPQR' de $var. */
echo substr_replace ($var, '', 10, -1) . "<br>\n";
?>
```

Vea también **str_replace()** y **substr()**.

> **Nota: substr_replace()** fue añadida en el PHP 4.0.

substr

(PHP 3, PHP 4 , PHP 5)

substr -- Devuelve parte de una cadena

Descripción

string **substr** (string cadena, int comienzo [, int largo])

substr devuelve la porción de *cadena* especificada por los parámetros *comienzo* y *largo*.

Si *comienzo* es positivo, la cadena devuelta comenzará en dicho caracter de *cadena*.

Ejemplos:

```
$resto = substr ("abcdef", 1);    // devuelve "bcdef"
$resto = substr ("abcdef", 1, 3); // devuelve "bcd"
```

Si *comienzo* es negativo, la cadena devuelta comenzará en dicha posición desde el final de *cadena*.

Ejemplos:

```
$resto = substr ("abcdef", -1);    // devuelve "f"
$resto = substr ("abcdef", -2);    // devuelve "ef"
$resto = substr ("abcdef", -3, 1); // devuelve "d"
```

Si se especifica *largo* y es positivo, la cadena devuelta terminará *largo* caracteres tras el *comienzo*. Si esto resulta en una cadena con longitud negativa (porque el comienzo está pasado el final de la cadena), la cadena devuelta contendrá únicamente el caracter que haya en *comienzo*.

Si se especifica *largo* y es negativo, la cadena devuelta terminará a *largo* caracteres desde el final de *cadena*. Si esto resulta en una cadena con longitud negativa, la cadena devuelta contendrá únicamente el caracter que haya en *comienzo*.

Examples:

```
$resto = substr ("abcdef", 1, -1); // devuelve "bcde"
```

Vea también **strrchr()**, **substr_replace()**, **ereg()**, y **trim()**.

trim

(PHP 3, PHP 4 , PHP 5)

trim -- Elimina espacios del principio y final de una cadena

Descripción

string **trim** (string cad)

Esta función elimina los espacios en blanco del comienzo y del final de una cadena y devuelve el resultado. Los caracteres de espacio que elimina realmente son: "\n", "\r", "\t", "\v", "\0", y el espacio en sí.

Vea también **chop()** y **ltrim()**.

ucfirst

(PHP 3, PHP 4 , PHP 5)

ucfirst -- Pasar a mayúsculas el primer caracter de una cadena

Descripción

string **ucfirst** (string cad)

Pone en mayúsculas el primer carácter de *cad* si es alfabético.

Nótese que 'alfabético' está determinado por la localidad actual. Por ejemplo, en la localidad por defecto "C", los caracteres como la a con diéresis (Ã*f*Â¤) no serán convertidos.

Ejemplo 1. Ejemplo de ucfirst()

```
$texto = 'susanita tiene un ratón, un ratón chiquitín.';
$texto = ucfirst ($texto); // $texto vale ahora: Susanita tiene un
                           // ratón, un ratón chiquitín.
```

Vea también **strtoupper()** y **strtolower()**

ucwords

(PHP 3>= 3.0.3, PHP 4 , PHP 5)

ucwords -- Pone en mayúsculas el primer caracter de cada palabra de una cadena

Descripción

string **ucwords** (string cad)

Pasa a mayúsculas la primera letra de cada palabra en *cad* si dicho caracter es alfabético.

Ejemplo 1. Ejemplo de ucwords()

```
$texto = "susanita tiene un ratón, un ratón chiquitín.";
$texto = ucwords($texto); // $texto vale ahora: Susanita Tiene Un
                          // Ratón, Un Ratón Chiquitín.
```

Vea también **strtoupper()**, **strtolower()** y **ucfirst()**.

vfprintf

(PHP 5)

vfprintf -- Write a formatted string to a stream

Description

int **vfprintf** (resource handle, string format, array args)

Write a string produced according to *format* to the stream resource specified by *handle*. *format* is described in the documentation for **sprintf()**.

Operates as **fprintf()** but accepts an array of arguments, rather than a variable number of arguments.

Returns the length of the outputted string.

See also: **printf()**, **sprintf()**, **sscanf()**, **fscanf()**, **vsprintf()**, and **number_format()**.

Examples

Ejemplo 1. vfprintf(): zero-padded integers

```
<?php
if (!($fp = fopen('date.txt', 'w')))
    return;

vfprintf($fp, "%04d-%02d-%02d", array($year, $month, $day));
// will write the formatted ISO date to date.txt
?>
```

vprintf

(PHP 4 >= 4.1.0, PHP 5)

vprintf -- Imprimir una cadena con formato

Descripción

int **vprintf** (string formato, array args)

Despliega los valores de una matriz como una cadena con formato de acuerdo al *formato* (el cual es descrito en la documentación de **sprintf()**).

Opera como **printf()** pero acepta una matriz como sus argumentos, en lugar de un número variable de argumentos.

Devuelve la longitud de la cadena impresa.

Vea también **printf()**, **sprintf()**, **vsprintf()**

vsprintf

(PHP 4 >= 4.1.0, PHP 5)

vsprintf -- Devuelve una cadena con formato

Descripción

string **vsprintf** (string formato, array args)

Devuelve los valores de una matriz como una cadena con formato de acuerdo a *formato* (el cual es descrito en la documentación de **sprintf()**).

Opera como **sprintf()** pero acepta una matriz de argumentos, en lugar de un número variable de argumentos.

Vea también **sprintf()** y **vprintf()**

wordwrap

(PHP 4 >= 4.0.2, PHP 5)

wordwrap -- Corta una cadena en un número dado de caracteres usando un caracter de ruptura de cadenas.

Descripción

string **wordwrap** (string cad [, int ancho [, string ruptura]])

Corta la cadena *cad* en la columna especificada por el parámetro (opcional) *ancho*. La línea se rompe utilizando el parámetro (opcional) *ruptura*.

wordwrap() automáticamente cortará en la columna 75 y usará '\n' (nueva línea) si no se especifican el *ancho* o la *ruptura*.

Ejemplo 1. Ejemplo de wordwrap()

```
$texto = "El veloz murciélago hindú comía feliz cardillo y kiwi.";
$textonuevo = wordwrap( $texto, 20 );

echo "$textonuevo\n";
```

Este ejemplo mostraría:

```
El veloz murciélago
hindú comía feliz cardillo y kiwi.
```

Vea también **nl2br()**.

CXXII. Shockwave Flash functions

PHP offers the ability to create Shockwave Flash files via Paul Haeberli's libswf module. You can download libswf at ftp://ftp.sgi.com/sgi/graphics/grafica/flash/. Once you have libswf all you need to do is to configure *--with-swf[=DIR]* where DIR is a location containing the directories include and lib. The include directory has to contain the swf.h file and the lib directory has to contain the libswf.a file. If you unpack the libswf distribution the two files will be in one directory. Consequently you will have to copy the files to the proper location manually.

Once you've successfully installed PHP with Shockwave Flash support you can then go about creating Shockwave files from PHP. You would be surprised at what you can do, take the following code:

Ejemplo 1. SWF example

```
<?php
swf_openfile ("test.swf", 256, 256, 30, 1, 1, 1);
swf_ortho2 (-100, 100, -100, 100);
swf_defineline (1, -70, 0, 70, 0, .2);
swf_definerect (4, 60, -10, 70, 0, 0);
swf_definerect (5, -60, 0, -70, 10, 0);
swf_addcolor (0, 0, 0, 0);

swf_definefont (10, "Mod");
swf_fontsize (5);
swf_fontslant (10);
swf_definetext (11, "This be Flash wit PHP!", 1);

swf_pushmatrix ();
swf_translate (-50, 80, 0);
swf_placeobject (11, 60);
swf_popmatrix ();

for ($i = 0; $i < 30; $i++) {
    $p = $i/(30-1);
    swf_pushmatrix ();
    swf_scale (1-($p*.9), 1, 1);
    swf_rotate (60*$p,  'z');
    swf_translate (20+20*$p, $p/1.5, 0);
    swf_rotate (270*$p,  'z');
    swf_addcolor ($p, 0, $p/1.2, -$p);
    swf_placeobject (1, 50);
    swf_placeobject (4, 50);
    swf_placeobject (5, 50);
    swf_popmatrix ();
    swf_showframe ();
}

for ($i = 0; $i < 30; $i++) {
    swf_removeobject (50);
    if (($i%4) == 0) {
        swf_showframe ();
    }
}

swf_startdoaction ();
swf_actionstop ();
swf_enddoaction ();

swf_closefile ();
?>
```

Nota: SWF support was added in PHP4 RC2.

Tabla de contenidos

swf_definepoly -- Define a polygon
swf_definerect -- Define a rectangle
swf_definetext -- Define a text string
swf_endbutton -- End the definition of the current button
swf_enddoaction -- End the current action
swf_endshape -- Completes the definition of the current shape
swf_endsymbol -- End the definition of a symbol
swf_fontsize -- Change the font size
swf_fontslant -- Set the font slant
swf_fonttracking -- Set the current font tracking
swf_getbitmapinfo -- Get information about a bitmap
swf_getfontinfo -- The height in pixels of a capital A and a lowercase x
swf_getframe -- Get the frame number of the current frame
swf_labelframe -- Label the current frame
swf_lookat -- Define a viewing transformation
swf_modifyobject -- Modify an object
swf_mulcolor -- Sets the global multiply color to the rgba value specified
swf_nextid -- Returns the next free object id
swf_oncondition -- Describe a transition used to trigger an action list
swf_openfile -- Open a new Shockwave Flash file
swf_ortho2 -- Defines 2D orthographic mapping of user coordinates onto the current viewport
swf_ortho -- Defines an orthographic mapping of user coordinates onto the current viewport
swf_perspective -- Define a perspective projection transformation
swf_placeobject -- Place an object onto the screen
swf_polarview -- Define the viewer's position with polar coordinates
swf_popmatrix -- Restore a previous transformation matrix
swf_posround -- Enables or Disables the rounding of the translation when objects are placed or moved
swf_pushmatrix -- Push the current transformation matrix back unto the stack
swf_removeobject -- Remove an object
swf_rotate -- Rotate the current transformation
swf_scale -- Scale the current transformation
swf_setfont -- Change the current font
swf_setframe -- Switch to a specified frame
swf_shapearc -- Draw a circular arc
swf_shapecurveto3 -- Draw a cubic bezier curve
swf_shapecurveto -- Draw a quadratic bezier curve between two points
swf_shapefillbitmapclip -- Set current fill mode to clipped bitmap
swf_shapefillbitmaptile -- Set current fill mode to tiled bitmap
swf_shapefilloff -- Turns off filling
swf_shapefillsolid -- Set the current fill style to the specified color
swf_shapelinesolid -- Set the current line style
swf_shapelineto -- Draw a line
swf_shapemoveto -- Move the current position
swf_showframe -- Display the current frame
swf_startbutton -- Start the definition of a button
swf_startdoaction -- Start a description of an action list for the current frame
swf_startshape -- Start a complex shape
swf_startsymbol -- Define a symbol
swf_textwidth -- Get the width of a string
swf_translate -- Translate the current transformations
swf_viewport -- Select an area for future drawing

swf_actiongeturl

(PHP 4)

swf_actiongeturl -- Get a URL from a Shockwave Flash movie

Description

void **swf_actiongeturl** (string url, string target)

The **swf_actionGetUrl()** function gets the URL specified by the parameter *url* with the target *target*.

swf_actiongotoframe

(PHP 4)

swf_actiongotoframe -- Play a frame and then stop

Description

void **swf_actiongotoframe** (int framenumber)

The **swf_actionGotoFrame()** function will go to the frame specified by *framenumber*, play it, and then stop.

swf_actiongotolabel

(PHP 4)

swf_actiongotolabel -- Display a frame with the specified label

Description

void **swf_actiongotolabel** (string label)

The **swf_actionGotoLabel()** function displays the frame with the label given by the *label* parameter and then stops.

swf_actionnextframe

(PHP 4)

swf_actionnextframe -- Go foward one frame

Description

void **swf_actionnextframe** (void)

Go foward one frame.

swf_actionplay

(PHP 4)

swf_actionplay -- Start playing the flash movie from the current frame

Description

void **swf_actionplay** (void)

Start playing the flash movie from the current frame.

swf_actionprevframe

(PHP 4)

swf_actionprevframe -- Go backwards one frame

Description

void **swf_actionprevframe** (void)

swf_actionsettarget

(PHP 4)

swf_actionsettarget -- Set the context for actions

Description

void **swf_actionsettarget** (string target)

The **swf_actionSetTarget()** function sets the context for all actions. You can use this to control other flash movies that are currently playing.

swf_actionstop

(PHP 4)

swf_actionstop -- Stop playing the flash movie at the current frame

Description

void **swf_actionstop** (void)

Stop playing the flash movie at the current frame.

swf_actiontogglequality

(PHP 4)

swf_actiontogglequality -- Toggle between low and high quality

Description

void **swf_actiontogglequality** (void)

Toggle the flash movie between high and low quality.

swf_actionwaitforframe

(PHP 4)

swf_actionwaitforframe -- Skip actions if a frame has not been loaded

Description

void **swf_actionwaitforframe** (int framenumber, int skipcount)

The **swf_actionWaitForFrame()** function will check to see if the frame, specified by the *framenumber* parameter has been loaded, if not it will skip the number of actions specified by the *skipcount* parameter. This can be useful for "Loading..." type animations.

swf_addbuttonrecord

(PHP 4)

swf_addbuttonrecord -- Controls location, appearance and active area of the current button

Description

void **swf_addbuttonrecord** (int states, int shapeid, int depth)

The **swf_addbuttonrecord()** function allows you to define the specifics of using a button. The first parameter, *states*, defines what states the button can have, these can be any or all of the following constants: BSHitTest, BSDown, BSOver or BSUp. The second parameter, the *shapeid* is the look of the button, this is usually the object id of the shape of the button. The *depth* parameter is the placement of the button in the current frame.

Ejemplo 1. swf_addbuttonrecord() function example

```
swf_startButton ($objid, TYPE_MENUBUTTON);
    swf_addButtonRecord (BSDown|BSOver, $buttonImageId, 340);
    swf_onCondition (MenuEnter);
        swf_actionGetUrl ("http://www.designmultimedia.com", "_level1");
    swf_onCondition (MenuExit);
        swf_actionGetUrl ("", "_level1");
swf_endButton ();
```

swf_addcolor

(PHP 4)

swf_addcolor -- Set the global add color to the rgba value specified

Description

void **swf_addcolor** (float r, float g, float b, float a)

The **swf_addcolor()** function sets the global add color to the *rgba* color specified. This color is then used (implicitly) by the **swf_placeobject()**, **swf_modifyobject()** and the **swf_addbuttonrecord()** functions. The color of the object will be add by the *rgba* values when the object is written to the screen.

> **Nota:** The *rgba* values can be either positive or negative.

swf_closefile

(PHP 4)

swf_closefile -- Close the current Shockwave Flash file

Description

void **swf_closefile** (void)

Close a file that was opened by the **swf_openfile()** function.

swf_definebitmap

(PHP 4)

swf_definebitmap -- Define a bitmap

Description

void **swf_definebitmap** (int objid, string image_name)

The **swf_definebitmap()** function defines a bitmap given a GIF, JPEG, RGB or FI image. The image will be converted into a Flash JPEG or Flash color map format.

swf_definefont

(PHP 4)

swf_definefont -- Defines a font

Description

void **swf_definefont** (int fontid, string fontname)

The **swf_definefont()** function defines a font given by the *fontname* parameter and gives it the id specified by the *fontid* parameter. It then sets the font given by *fontname* to the current font.

swf_defineline

(PHP 4)

swf_defineline -- Define a line

Description

void **swf_defineline** (int objid, float x1, float y1, float x2, float y2, float width)

The **swf_defineline()** defines a line starting from the x coordinate given by *x1* and the y coordinate given by *y1* parameter. Up to the x coordinate given by the *x2* parameter and the y coordinate given by the *y2* parameter. It will have a width defined by the *width* parameter.

swf_definepoly

(PHP 4)

swf_definepoly -- Define a polygon

Description

void **swf_definepoly** (int objid, array coords, int npoints, float width)

The **swf_definepoly()** function defines a polygon given an array of x, y coordinates (the coordinates are defined in the parameter *coords*). The parameter *npoints* is the number of overall points that are contained in the array given by *coords*. The *width* is the width of the polygon's border, if set to 0.0 the polygon is filled.

swf_definerect

(PHP 4)

swf_definerect -- Define a rectangle

Description

void **swf_definerect** (int objid, float x1, float y1, float x2, float y2, float width)

The **swf_definerect()** defines a rectangle with an upper left hand coordinate given by the x, *x1*, and the y, *y1*. And a lower right hand coordinate given by the x coordinate, *x2*, and the y coordinate, *y2* . Width of the rectangles border is given by the *width* parameter, if the width is 0.0 then the rectangle

is filled.

swf_definetext

(PHP 4)

swf_definetext -- Define a text string

Description

void **swf_definetext** (int objid, string str, int docenter)

Define a text string (the *str* parameter) using the current font and font size. The *docenter* is where the word is centered, if *docenter* is 1, then the word is centered in x.

swf_endbutton

(PHP 4)

swf_endbutton -- End the definition of the current button

Description

void **swf_endbutton** (void)

The **swf_endButton()** function ends the definition of the current button.

swf_enddoaction

(PHP 4)

swf_enddoaction -- End the current action

Description

void **swf_enddoaction** (void)

Ends the current action started by the **swf_startdoaction()** function.

swf_endshape

(PHP 4)

swf_endshape -- Completes the definition of the current shape

Description

void **swf_endshape** (void)

The **swf_endshape()** completes the definition of the current shape.

swf_endsymbol

(PHP 4)

swf_endsymbol -- End the definition of a symbol

Description

void **swf_endsymbol** (void)

The **swf_endsymbol()** function ends the definition of a symbol that was started by the **swf_startsymbol()** function.

swf_fontsize

(PHP 4)

swf_fontsize -- Change the font size

Description

void **swf_fontsize** (float size)

The **swf_fontsize()** function changes the font size to the value given by the *size* parameter.

swf_fontslant

(PHP 4)

swf_fontslant -- Set the font slant

Description

void **swf_fontslant** (float slant)

Set the current font slant to the angle indicated by the *slant* parameter. Positive values create a foward slant, negative values create a negative slant.

swf_fonttracking

(PHP 4)

swf_fonttracking -- Set the current font tracking

Description

void **swf_fonttracking** (float tracking)

Set the font tracking to the value specified by the *tracking* parameter. This function is used to increase the spacing between letters and text, positive values increase the space and negative values decrease the space between letters.

swf_getbitmapinfo

(PHP 4)

swf_getbitmapinfo -- Get information about a bitmap

Description

array **swf_getbitmapinfo** (int bitmapid)

The **swf_getbitmapinfo()** function returns an array of information about a bitmap given by the *bitmapid* parameter. The returned array has the following elements:

- "size" - The size in bytes of the bitmap.
- "width" - The width in pixels of the bitmap.
- "height" - The height in pixels of the bitmap.

swf_getfontinfo

(PHP 4)

swf_getfontinfo -- The height in pixels of a capital A and a lowercase x

Description

array **swf_getfontinfo** (void)

The **swf_getfontinfo()** function returns an associative array with the following parameters:

- Aheight - The height in pixels of a capital A.
- xheight - The height in pixels of a lowercase x.

swf_getframe

(PHP 4)

swf_getframe -- Get the frame number of the current frame

Description

int **swf_getframe** (void)

The **swf_getframe()** function gets the number of the current frame.

swf_labelframe

(PHP 4)

swf_labelframe -- Label the current frame

Description

void **swf_labelframe** (string name)

Label the current frame with the name given by the *name* parameter.

swf_lookat

(PHP 4)

swf_lookat -- Define a viewing transformation

Description

void **swf_lookat** (double view_x, double view_y, double view_z, double reference_x, double reference_y, double reference_z, double twist)

The **swf_lookat()** function defines a viewing transformation by giving the viewing position (the parameters *view_x*, *view_y*, and *view_z*) and the coordinates of a reference point in the scene, the reference point is defined by the *reference_x*, *reference_y* , and *reference_z* parameters. The *twist* controls the rotation along with viewer's z axis.

swf_modifyobject

(PHP 4)

swf_modifyobject -- Modify an object

Description

void **swf_modifyobject** (int depth, int how)

Updates the position and/or color of the object at the specified depth, *depth*. The parameter *how* determines what is updated. *how* can either be the constant MOD_MATRIX or MOD_COLOR or it can be a combination of both (MOD_MATRIX|MOD_COLOR).

MOD_COLOR uses the current mulcolor (specified by the function **swf_mulcolor()**) and addcolor

(specified by the function **swf_addcolor()**) to color the object. MOD_MATRIX uses the current matrix to position the object.

swf_mulcolor

(PHP 4)

swf_mulcolor -- Sets the global multiply color to the rgba value specified

Description

void **swf_mulcolor** (float r, float g, float b, float a)

The **swf_mulcolor()** function sets the global multiply color to the *rgba* color specified. This color is then used (implicitly) by the **swf_placeobject()**, **swf_modifyobject()** and the **swf_addbuttonrecord()** functions. The color of the object will be multiplied by the *rgba* values when the object is written to the screen.

> **Nota:** The *rgba* values can be either positive or negative.

swf_nextid

(PHP 4)

swf_nextid -- Returns the next free object id

Description

int **swf_nextid** (void)

The **swf_nextid()** function returns the next available object id.

swf_oncondition

(PHP 4)

swf_oncondition -- Describe a transition used to trigger an action list

Description

void **swf_oncondition** (int transition)

The **swf_onCondition()** function describes a transition that will trigger an action list. There are several types of possible transitions, the following are for buttons defined as TYPE_MENUBUTTON:

- IdletoOverUp
- OverUptoIdle

- OverUptoOverDown
- OverDowntoOverUp
- IdletoOverDown
- OutDowntoIdle
- MenuEnter (IdletoOverUp|IdletoOverDown)
- MenuExit (OverUptoIdle|OverDowntoIdle)

For TYPE_PUSHBUTTON there are the following options:

- IdletoOverUp
- OverUptoIdle
- OverUptoOverDown
- OverDowntoOverUp
- OverDowntoOutDown
- OutDowntoOverDown
- OutDowntoIdle
- ButtonEnter (IdletoOverUp|OutDowntoOverDown)
- ButtonExit (OverUptoIdle|OverDowntoOutDown)

swf_openfile

(PHP 4)

swf_openfile -- Open a new Shockwave Flash file

Description

void **swf_openfile** (string filename, float width, float height, float framerate, float r, float g, float b)

The **swf_openfile()** function opens a new file named *filename* with a width of *width* and a height of *height* a frame rate of *framerate* and background with a red color of *r* a green color of *g* and a blue color of *b*.

The **swf_openfile()** must be the first function you call, otherwise your script will cause a segfault. If you want to send your output to the screen make the filename: "php://stdout" (support for this is in 4.0.1 and up).

swf_ortho2

(PHP 4)

swf_ortho2 -- Defines 2D orthographic mapping of user coordinates onto the current viewport

Description

void **swf_ortho2** (double xmin, double xmax, double ymin, double ymax)

The **swf_ortho2()** function defines a two dimensional orthographic mapping of user coordinates onto the current viewport, this defaults to one to one mapping of the area of the Flash movie. If a perspective transformation is desired, the **swf_perspective ()** function can be used.

swf_ortho

(PHP 4 >= 4.0.1)

swf_ortho -- Defines an orthographic mapping of user coordinates onto the current viewport

Description

void **swf_ortho** (double xmin, double xmax, double ymin, double ymax, double zmin, double zmax)

The **swf_ortho()** funcion defines a orthographic mapping of user coordinates onto the current viewport.

swf_perspective

(PHP 4)

swf_perspective -- Define a perspective projection transformation

Description

void **swf_perspective** (double fovy, double aspect, double near, double far)

The **swf_perspective()** function defines a perspective projection transformation. The *fovy* parameter is field-of-view angle in the y direction. The *aspect* parameter should be set to the aspect ratio of the viewport that is being drawn onto. The *near* parameter is the near clipping plane and the *far* parameter is the far clipping plane.

> **Nota:** Various distortion artifacts may appear when performing a perspective projection, this is because Flash players only have a two dimensional matrix. Some are not to pretty.

swf_placeobject

(PHP 4)

swf_placeobject -- Place an object onto the screen

Description

void **swf_placeobject** (int objid, int depth)

Places the object specified by *objid* in the current frame at a depth of *depth*. The *objid* parameter and the *depth* must be between 1 and 65535.

This uses the current mulcolor (specified by **swf_mulcolor()**) and the current addcolor (specified by **swf_addcolor()**) to color the object and it uses the current matrix to position the object.

> **Nota:** Full RGBA colors are supported.

swf_polarview

(PHP 4)

swf_polarview -- Define the viewer's position with polar coordinates

Description

void **swf_polarview** (double dist, double azimuth, double incidence, double twist)

The **swf_polarview()** function defines the viewer's position in polar coordinates. The *dist* parameter gives the distance between the viewpoint to the world space origin. The *azimuth* parameter defines the azimuthal angle in the x,y coordinate plane, measured in distance from the y axis. The *incidence* parameter defines the angle of incidence in the y,z plane, measured in distance from the z axis. The incidence angle is defined as the angle of the viewport relative to the z axis. Finally the *twist* specifies the amount that the viewpoint is to be rotated about the line of sight using the right hand rule.

swf_popmatrix

(PHP 4)

swf_popmatrix -- Restore a previous transformation matrix

Description

void **swf_popmatrix** (void)

The **swf_popmatrix()** function pushes the current transformation matrix back onto the stack.

swf_posround

(PHP 4)

swf_posround -- Enables or Disables the rounding of the translation when objects are placed or moved

Description

void **swf_posround** (int round)

The **swf_posround()** function enables or disables the rounding of the translation when objects are placed or moved, there are times when text becomes more readable because rounding has been enabled. The *round* is whether to enable rounding or not, if set to the value of 1, then rounding is enabled, if set to 0 then rounding is disabled.

swf_pushmatrix

(PHP 4)

swf_pushmatrix -- Push the current transformation matrix back unto the stack

Description

void **swf_pushmatrix** (void)

The **swf_pushmatrix()** function pushes the current transformation matrix back onto the stack.

swf_removeobject

(PHP 4)

swf_removeobject -- Remove an object

Description

void **swf_removeobject** (int depth)

Removes the object at the depth specified by *depth*.

swf_rotate

(PHP 4)

swf_rotate -- Rotate the current transformation

Description

void **swf_rotate** (double angle, string axis)

The **swf_rotate()** rotates the current transformation by the angle given by the *angle* parameter around the axis given by the *axis* parameter. Valid values for the axis are 'x' (the x axis), 'y' (the y axis) or 'z' (the z axis).

swf_scale

(PHP 4)

swf_scale -- Scale the current transformation

Description

void **swf_scale** (double x, double y, double z)

The **swf_scale()** scales the x coordinate of the curve by the value of the *x* parameter, the y coordinate of the curve by the value of the *y* parameter, and the z coordinate of the curve by the value of the *z* parameter.

swf_setfont

(PHP 4)

swf_setfont -- Change the current font

Description

void **swf_setfont** (int fontid)

The **swf_setfont()** sets the current font to the value given by the *fontid* parameter.

swf_setframe

(PHP 4)

swf_setframe -- Switch to a specified frame

Description

void **swf_setframe** (int framenumber)

The **swf_setframe()** changes the active frame to the frame specified by *framenumber*.

swf_shapearc

(PHP 4)

swf_shapearc -- Draw a circular arc

Description

void **swf_shapearc** (float x, float y, float r, float ang1, float ang2)

The **swf_shapeArc()** function draws a circular arc from angle A given by the *ang1* parameter to angle B given by the *ang2* parameter. The center of the circle has an x coordinate given by the *x* parameter and a y coordinate given by the *y*, the radius of the circle is given by the *r* parameter.

swf_shapecurveto3

(PHP 4)

swf_shapecurveto3 -- Draw a cubic bezier curve

Description

void **swf_shapecurveto3** (float x1, float y1, float x2, float y2, float x3, float y3)

Draw a cubic bezier curve using the x,y coordinate pairs *x1*, *y1* and *x2*,*y2* as off curve control points and the x,y coordinate *x3*, *y3* as an endpoint. The current position is then set to the x,y coordinate pair given by *x3*,*y3*.

swf_shapecurveto

(PHP 4)

swf_shapecurveto -- Draw a quadratic bezier curve between two points

Description

void **swf_shapecurveto** (float x1, float y1, float x2, float y2)

The **swf_shapecurveto()** function draws a quadratic bezier curve from the x coordinate given by *x1* and the y coordinate given by *y1* to the x coordinate given by *x2* and the y coordinate given by *y2*. The current position is then set to the x,y coordinates given by the *x2* and *y2* parameters

swf_shapefillbitmapclip

(PHP 4)

swf_shapefillbitmapclip -- Set current fill mode to clipped bitmap

Description

void **swf_shapefillbitmapclip** (int bitmapid)

Sets the fill to bitmap clipped, empty spaces will be filled by the bitmap given by the *bitmapid* parameter.

swf_shapefillbitmaptile

(PHP 4)

swf_shapefillbitmaptile -- Set current fill mode to tiled bitmap

Description

void **swf_shapefillbitmaptile** (int bitmapid)

Sets the fill to bitmap tile, empty spaces will be filled by the bitmap given by the *bitmapid* parameter (tiled).

swf_shapefilloff

(PHP 4)

swf_shapefilloff -- Turns off filling

Description

void **swf_shapefilloff** (void)

The **swf_shapeFillOff()** function turns off filling for the current shape.

swf_shapefillsolid

(PHP 4)

swf_shapefillsolid -- Set the current fill style to the specified color

Description

void **swf_shapefillsolid** (float r, float g, float b, float a)

The **swf_shapeFillSolid()** function sets the current fill style to solid, and then sets the fill color to the values of the *rgba* parameters.

swf_shapelinesolid

(PHP 4)

swf_shapelinesolid -- Set the current line style

Description

void **swf_shapelinesolid** (float r, float g, float b, float a, float width)

The **swf_shapelinesolid()** function sets the current line style to the color of the *rgba* parameters and width to the *width* parameter. If 0.0 is given as a width then no lines are drawn.

swf_shapelineto

(PHP 4)

swf_shapelineto -- Draw a line

Description

void **swf_shapelineto** (float x, float y)

The **swf_shapeLineTo()** draws a line to the x,y coordinates given by the *x* parameter & the *y* parameter. The current position is then set to the x,y parameters.

swf_shapemoveto

(PHP 4)

swf_shapemoveto -- Move the current position

Description

void **swf_shapemoveto** (float x, float y)

The **swf_shapeMoveTo()** function moves the current position to the x coordinate given by the *x* parameter and the y position given by the *y* parameter.

swf_showframe

(PHP 4)

swf_showframe -- Display the current frame

Description

void **swf_showframe** (void)

The swf_showframe function will output the current frame.

swf_startbutton

(PHP 4)

swf_startbutton -- Start the definition of a button

Description

void **swf_startbutton** (int objid, int type)

The **swf_startbutton()** function starts off the definition of a button. The *type* parameter can either be TYPE_MENUBUTTON or TYPE_PUSHBUTTON. The TYPE_MENUBUTTON constant allows the focus to travel from the button when the mouse is down, TYPE_PUSHBUTTON does not allow the focus to travel when the mouse is down.

swf_startdoaction

(PHP 4)

swf_startdoaction -- Start a description of an action list for the current frame

Description

void **swf_startdoaction** (void)

The **swf_startdoaction()** function starts the description of an action list for the current frame. This must be called before actions are defined for the current frame.

swf_startshape

(PHP 4)

swf_startshape -- Start a complex shape

Description

void **swf_startshape** (int objid)

The **swf_startshape()** function starts a complex shape, with an object id given by the *objid* parameter.

swf_startsymbol

(PHP 4)

swf_startsymbol -- Define a symbol

Description

void **swf_startsymbol** (int objid)

Define an object id as a symbol. Symbols are tiny flash movies that can be played simultaneously. The *objid* parameter is the object id you want to define as a symbol.

swf_textwidth

(PHP 4)

swf_textwidth -- Get the width of a string

Description

float **swf_textwidth** (string str)

The **swf_textwidth()** function gives the width of the string, *str*, in pixels, using the current font and font size.

swf_translate

(PHP 4)

swf_translate -- Translate the current transformations

Description

void **swf_translate** (double x, double y, double z)

The **swf_translate()** function translates the current transformation by the *x*, *y*, and *z* values given.

swf_viewport

(PHP 4)

swf_viewport -- Select an area for future drawing

Description

void **swf_viewport** (double xmin, double xmax, double ymin, double ymax)

The **swf_viewport()** function selects an area for future drawing for *xmin* to *xmax* and *ymin* to *ymax*, if this function is not called the area defaults to the size of the screen.

CXXIII. Funciones de Sybase

Tabla de contenidos

sybase_affected_rows -- Obtiene el número de filas afectadas en la última consulta
sybase_close -- Cierra una conexión Sybase
sybase_connect -- Abre una conexión con un servidor Sybase
sybase_data_seek -- Mueve el puntero interno de la fila
sybase_deadlock_retry_count -- Fija el contador de reintentos del deadlock
sybase_fetch_array -- Obtiene una fila como una matriz
sybase_fetch_assoc -- Obtiene el resultado de la sentencia como una matriz asociativa
sybase_fetch_field -- obtiene la información del campo
sybase_fetch_object -- carga una fila como un objeto
sybase_fetch_row -- obtiene una fila como una matriz enumerada
sybase_field_seek -- establece el offset de un campo
sybase_free_result -- libera el resultado de la memoria
sybase_get_last_message -- Regresa el último mensaje del servidor Sybase
sybase_min_client_severity -- Fija el nivel minimo de severidad del problema reportado por el servidor
sybase_min_error_severity -- Fija el error minimo a reportar según su severidad
sybase_min_message_severity -- Fija el mensaje Sets minimum message severity
sybase_min_server_severity -- Fija el nivel minimo de severidad del problema reportado por el servidor
sybase_num_fields -- obtiene el número de campos de un resultado
sybase_num_rows -- obtiene el número de filas de un resultado
sybase_pconnect -- abre una conexión persistente con Sybase
sybase_query -- envía una consulta a Sybase
sybase_result -- obtiene datos de un resultado
sybase_select_db -- seleccciona una base de datos Sybase
sybase_set_message_handler -- Fija el manejador a ser llamado cuando se emite un mensaje del sistema
sybase_unbuffered_query -- Envía una sentencia SQL a Sybase sin bloquear

sybase_affected_rows

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

sybase_affected_rows -- Obtiene el número de filas afectadas en la última consulta

Descripción

int **sybase_affected_rows** ([resource id_enlace])

sybase_affected_rows() devuelve el número de filas afectadas por la última consulta INSERT, UPDATE o DELETE en el servidor asociado con el identificador de enlace dado. Si no se especifica el identificador de enlace, se asume el último enlace abierto.

Ejemplo 1. Consulta-Delete

```
<?php
    /* conectarse con la base de datos */
    sybase_connect('SYBASE', '', '') or
        die("No pudo conectarse");
    sybase_select_db("bd");

    sybase_query("DELETE FROM alguna_tabla WHERE id < 10");
    printf("Registros eliminados: %d\n", sybase_affected_rows());
?>
```

El anterior ejemplo produciría la siguiente salida:

```
Registros eliminados: 10
```

Este comando no es efectivo para sentencias SELECT, únicamente en sentencias que modifican registros. Para recuperar el número de filas devueltas desde un SELECT, use **sybase_num_rows()**.

> **Nota:** Esta función se encuentra disponible solamente cuando se usan las bibliotecas CT de Sybase y no las bibliotecas DB.

Vea también **sybase_num_rows()**.

sybase_close

(PHP 3, PHP 4 , PHP 5)

sybase_close -- Cierra una conexión Sybase

Descripción

bool **sybase_close** ([int identificador_de_enlace])

sybase_close() cierra el enlace a una base de datos Sybase que está asociada con el enlace *identificador_de_enlace*. Si el identificador de enlace no se especifica, se asume el último enlace abierto.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Note que esto no es usualmente necesario, dado que los enlaces abiertos como no persistentes son cerradoa automáticamente al final de la ejecución del script.

sybase_close() no cierra enlaces persistentes generados por **sybase_pconnect()**.

Vea también **sybase_connect()**, **sybase_pconnect()**.

sybase_connect

(PHP 3, PHP 4 , PHP 5)

sybase_connect -- Abre una conexión con un servidor Sybase

Descripción

resource **sybase_connect** ([string nombre_servidor [, string nombre_usuario [, string contrasenya [, string juego_caracteres [, string nombre_aplicacion]]]]])

Devuelve un identificador de enlace Sybase positivo de tener éxito, o **FALSE** si fracasa.

sybase_connect() establece una conexión con un servidor Sybase. El argumento nombre_servidor tiene que ser un nombre de servidor válido que esté definido en el archivo 'interfaces'.

En caso de que se realice una segunda llamada a **sybase_connect()** con los mismos argumentos, no se establece un nuevo enlace, en su lugar, se devolverá el identificador del enlace ya abierto.

El enlace con el servidor será cerrado tan pronto como finalice la ejecución del script, a menos que sea cerrado antes al llamar explícitamente **sybase_close()**.

Ejemplo 1. Ejemplo de sybase_connect()

```
<?php
    $enlace = sybase_connect('SYBASE', '', '')
              or die("&iexcl;No pudo conectarse!");
    echo "Se conect&oacute; satisfactoriamnete";
    sybase_close($enlace);
?>
```

Vea también **sybase_pconnect()** y **sybase_close()**.

sybase_data_seek

(PHP 3, PHP 4 , PHP 5)

sybase_data_seek -- Mueve el puntero interno de la fila

Descripción

int **sybase_data_seek** (int result_identifier, int row_number)

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

sybase_data_seek() mueve el puntero interno de la fila del resultado asociado con el identificador de resultado específicado hacia el número de fila introducido. La siguiente llamada a **sybase_fetch_row()** devolverá esa fila.

Vea también **sybase_data_seek()**.

sybase_deadlock_retry_count

(PHP 4 >= 4.3.0, PHP 5)

sybase_deadlock_retry_count -- Fija el contador de reintentos del deadlock

Descripción

void **sybase_deadlock_retry_count** (int contador_de_reintentos)

Usando **sybase_deadlock_retry_count()**, el número de reintentos puede ser definido en caso de encontrarse en deadlock. Por defecto, cada deadlock sigue intentando un número infinito de veces o hasta que el proceso es destruido por Sybase, el script es finalizado (por ejemplo, por **set_time_limit()**) ó la sentencia SQL es exitosa.

> **Nota:** Esta función se encuentra disponible solamente cuando se usan las bibliotecas CT de Sybase y no las bibliotecas DB.

Tabla 1. Valores para contador_de_reintentos

-1	Reintenta por siempre (por defecto)
0	No reintenta
n	Reintenta n veces

sybase_fetch_array

(PHP 3, PHP 4 , PHP 5)

sybase_fetch_array -- Obtiene una fila como una matriz

Descripción

matriz **sybase_fetch_array** (resource result)

Regresa una matriz que corresponde a la fila obtenida, o **FALSE** si no hay más filas.

sybase_fetch_array() es una versión extendida de **sybase_fetch_row()**. Además de almacenar los datos en índices numéricos en la matriz resultante, tambiÃ‚Â´n almacena los datos en índices asociativos, usando los nombres de los campos como llaves.

Una cosa importate es que usando **sybase_fetch_array()** no es significativamente más lento que usar **sybase_fetch_row()**, mientras que si provee un valor agregado significativo.

> **Nota:** Cuando se seleccionan campos con nombres idénticos (por ejemplo, en una unión), los índices asociativos tendrán agregado un número secuencial, vea el ejemplo para más detalles.

Ejemplo 1. Nombres de campo idénticos

```
<?php
    $dbh = sybase_connect('SYBASE', '', '');
    $q = sybase_query('SELECT * FROM p, a WHERE p.person_id= a.person_id');
    var_dump(sybase_fetch_array($q));
    sybase_close($dbh);
?>
```

El ejemplo anterior producirá la siguiente salida (asumiendo que las dos tablas solo tienen una columna llamada "person_id" cada una):

```
array(4) {
  [0]=>
  int(1)
  ["person_id"]=>
  int(1)
  [1]=>
  int(1)
  ["person_id1"]=>
  int(1)
}
```

Vea también **sybase_fetch_row()**, **sybase_fetch_assoc()**, **sybase_fetch_object()**.

sybase_fetch_assoc

(PHP 4 >= 4.3.0, PHP 5)

sybase_fetch_assoc -- Obtiene el resultado de la sentencia como una matriz asociativa

Descripción

matriz **sybase_fetch_assoc** (int resultado)

Regresa una matriz que corresponde a las filas obtenidas o **FALSE** si no hay mas filas.

> **Nota:** Esta función se encuentra disponible solamente cuando se usan las bibliotecas CT de Sybase y no las bibliotecas DB.

sybase_fetch_assoc() es una adaptación de **sybase_fetch_row()** que usa los nombre de las columnas en lugar de números enteros como indices en la matriz resultante. En caso de columnas de diferentes tablas, con el mismo nombre, éstas son expresadas como: nombre, nombre1, nombre2, ..., nombreN.

Algo por tener en cuenta es, que usando **sybase_fetch_assoc()** significativamente no es más lenta que usar **sybase_fetch_row()**, dado que provee un valor agregado.

Vea también **sybase_fetch_array()**, **sybase_fetch_object()** y **sybase_fetch_row()**.

sybase_fetch_field

(PHP 3, PHP 4 , PHP 5)

sybase_fetch_field -- obtiene la información del campo

Descripción

object **sybase_fetch_field** (int result [, int field_offset])

Devuelve un objeto conteniendo la información del campo

sybase_fetch_field() puede usarse para obtener información sobre los campos de una consulta determinada. Si no se específica el offset del campo, el siguiente campo que aún no halla sido tomado por **sybase_fetch_field()** es el que se obtiene.

Las propiedades del objeto son:

- name - nombre de la columna. si la columna es el resultado de una función, esta propiedad se establece a computed#N, donde #N es un número de serie.
- column_source - la tabla de la cual se tomo la columna
- max_length - máxima longitud de la columna
- numeric - 1 si la columna es numérica

Vea también **sybase_field_seek()**.

sybase_fetch_object

(PHP 3, PHP 4 , PHP 5)

sybase_fetch_object -- carga una fila como un objeto

Descripción

int **sybase_fetch_object** (int result [, mixed object])

Devuelve un objeto con las propiedades que corresponden a la fila tomada, o **FALSE** si no hay más filas.

sybase_fetch_object() es similar a **sybase_fetch_assoc()**, con una diferencia - se devuelve un objeto.

Use el segundo *object* para específicar el tipo del objeto que quiere de regreso. Si este parámetro es omitido, el objeto será de tipo stdClass.

> **Nota:** A partir de PHP 4.3.0, esta funció no regresará miembros de objetos numéricos.
>
> Comportamiento anterior:
> ```
> object(stdclass)(3) {
>   [0]=>
>   string(3) "foo"
>   ["foo"]=>
>   string(3) "foo"
>   [1]=>
>   string(3) "bar"
>   ["bar"]=>
>   string(3) "bar"
> }
> ```
> Nuevo comportamiento:
> ```
> object(stdclass)(3) {
>   ["foo"]=>
>   string(3) "foo"
>   ["bar"]=>
>   string(3) "bar"
> }
> ```

Ejemplo 1. sybase_fetch_object() regresa Foo

```
<?php
    class Foo {
        var $foo, $bar, $baz;
    }

    // {...]
    $qrh= sybase_query('SELECT foo, bar, baz FROM example');
    $foo= sybase_fetch_object($qrh, 'Foo');
    $bar= sybase_fetch_object($qrh, new Foo());
    // {...]
?>
```

En términos de velocidad, esta función es identica a **sybase_fetch_array()**, y casi tan rápida como **sybase_fetch_row()** (la diferencia es insignificante).

Vea también **sybase_fetch_array()**, **sybase_fetch_row()**.

sybase_fetch_row

(PHP 3, PHP 4 , PHP 5)

sybase_fetch_row -- obtiene una fila como una matriz enumerada

Descripción

array **sybase_fetch_row** (int result)

Devuelve una matriz que corresponde a la fila obtenida, o **FALSE** si no hay más filas.

sybase_fetch_row() obtiene una fila de datos del resultado asociado con el identificador de resultado específicado. La fila se devuelve como una matriz. Cada columna del resultado es almacenada en un offset de la matriz, comenzando en el offset 0.

Las siguientes llamadas a **sybase_fetch_row()** devolverán la siguiente fila del resultado, o **FALSE** si no hay más filas.

Tabla 1. Tipos de Datos

PHP	Sybase
string	VARCHAR, TEXT, CHAR, IMAGE, BINARY, VARBINARY, DATETIME
int	NUMERIC (w/o precision), DECIMAL (w/o precision), INT, BIT, TINYINT, SMALLINT
float	NUMERIC (w/ precision), DECIMAL (w/ precision), REAL, FLOAT, MONEY
NULL	NULL

Vea también **sybase_fetch_array()**, **sybase_fetch_assoc()**, **sybase_data_seek()**, **sybase_fetch_object()**, **sybase_result()**.

sybase_field_seek

(PHP 3, PHP 4 , PHP 5)

sybase_field_seek -- establece el offset de un campo

Descripción

int **sybase_field_seek** (int result, int field_offset)

Localiza el campo específicado por el offset. Si la siguiente llamada **sybase_fetch_field()** no incluye un offset se devuelve este campo.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Vea también **sybase_fetch_field()**.

sybase_free_result

(PHP 3, PHP 4 , PHP 5)

sybase_free_result -- libera el resultado de la memoria

Descripción

int **sybase_free_result** (int result)

sybase_free_result() sólo se necesita usar en el caso de que este preocupado por el uso de demasiada memoria mientras se ejecuta su script. Todos los resultados en memoria son liberados cuando el script finaliza, puede llamar a **sybase_free_result()** con el identificador de resultado como argumento y la memoria asociada a ese resultado será liberada.

sybase_get_last_message

(PHP 3, PHP 4 , PHP 5)

sybase_get_last_message -- Regresa el último mensaje del servidor Sybase

Descripción

cadena **sybase_get_last_message** (void)

sybase_get_last_message() Regresa el último mensaje reportado por el servidor Sybase.

Vea también **sybase_min_message_severity()**.

sybase_min_client_severity

(PHP 3, PHP 4 , PHP 5)

sybase_min_client_severity -- Fija el nivel minimo de severidad del problema reportado por el servidor

Descripción

void **sybase_min_client_severity** (int severidad_del_problema)

sybase_min_client_severity() Fija el minimo nivel de severidad a reportar del tipo de falla que el servidor haya encontrado.

> **Nota:** Esta función se encuentra disponible solamente cuando se usan las bibliotecas CT de Sybase y no las bibliotecas DB.

Vea también **sybase_min_server_severity()**.

sybase_min_error_severity

(PHP 3, PHP 4 , PHP 5)

sybase_min_error_severity -- Fija el error minimo a reportar según su severidad

Descripción

void **sybase_min_error_severity** (int severidad)

sybase_min_error_severity() Fije el nivel minimo de error a reportar según su severidad.

> **Nota:** Esta función se encuentra disponible solamente cuando se usan las bibliotecas CT de Sybase y no las bibliotecas DB.

Vea también **sybase_min_message_severity()**.

sybase_min_message_severity

(PHP 3, PHP 4 , PHP 5)

sybase_min_message_severity -- Fija el mensaje Sets minimum message severity

Descripción

void **sybase_min_message_severity** (int severidad)

sybase_min_message_severity() Fija el nivel minimo del mensaje de error según su severidad.

> **Nota:** Esta función se encuentra disponible solamente cuando se usan las bibliotecas CT de Sybase y no las bibliotecas DB.

Vea también **sybase_min_error_severity()**.

sybase_min_server_severity

(PHP 3, PHP 4 , PHP 5)

sybase_min_server_severity -- Fija el nivel minimo de severidad del problema reportado por el servidor

Descripción

void **sybase_min_server_severity** (int severidad_del_problema)

sybase_min_server_severity() Fija el minimo nivel de severidad a reportar del tipo de falla que el servidor haya encontrado.

> **Nota:** Esta función se encuentra disponible solamente cuando se usan las bibliotecas CT

de Sybase y no las bibliotecas DB.

Vea también **sybase_min_client_severity()**.

sybase_num_fields

(PHP 3, PHP 4 , PHP 5)

sybase_num_fields -- obtiene el número de campos de un resultado

Descripción

int **sybase_num_fields** (int result)

sybase_num_fields() devuelve el número de campos en un resultado.

Vea también **sybase_query()**, **sybase_fetch_field()**, **sybase_num_rows()**.

sybase_num_rows

(PHP 3, PHP 4 , PHP 5)

sybase_num_rows -- obtiene el número de filas de un resultado

Descripción

int **sybase_num_rows** (string result)

sybase_num_rows() devuelve el número de filas de un resultado.

Vea también: **sybase_num_fields()**, **sybase_query()**, **sybase_fetch_row()**.

sybase_pconnect

(PHP 3, PHP 4 , PHP 5)

sybase_pconnect -- abre una conexión persistente con Sybase

Descripción

int **sybase_pconnect** ([string servername [, string username [, string password [, string charset [, string appname]]]]])

Devuelve: Un identificador de enlace persistente de Sybase positivo en caso de que pueda abrirlo, en caso de error devuelve **`FALSE`**

sybase_pconnect() actúa de forma muy parecida a **sybase_connect()** con dos grandes diferencias.

Primera, cuando se conecta, esta función primero tratará de encontrar un enlace (persistente) que ya

esté abierto con el mismo host, nombre de usuario y contraseña. Si encuentra uno, devolverá un identificador para él en vez de abrir una nueva conexión.

Segunda, la conexión al servidor SQL no se cerrará cuando finalice la ejecución del script. En vez de esto, el enlace permanecerá abierto para un futuro uso (**sybase_close()** no podrá cerrar enlaces establecidos con*sybase_pconnect()*).

Este tipo de enlaces son llamados 'persistentes'.

Vea también **sybase_connect()**.

sybase_query

(PHP 3, PHP 4 , PHP 5)

sybase_query -- envía una consulta a Sybase

Descripción

int **sybase_query** (string query [, int identificador_de_enlace])

Devuelve un identificador de resultado Sybase positivo si es exitoso, o **FALSE** en caso de error.

sybase_query() envía una consulta a la actual base de datos activa en el servidor que está asociada con el identificador de enlace específicado. Si no se específica un identificador de enlace, se asume el último enlace abierto. Si no hay un enlace abierto, la función intentará establecer un enlace como si **sybase_connect()** fuese llamada, y lo usará.

Vea también: **sybase_select_db()**, **sybase_connect()**.

sybase_result

(PHP 3, PHP 4 , PHP 5)

sybase_result -- obtiene datos de un resultado

Descripción

int **sybase_result** (int result, int i, mixed field)

Devuelve el contenido de la celda en la fila y el offset específicado de un conjunto de resultados de Sybase.

sybase_result() devuelve el contenido de una celda de un resultado de Sybase. El parámetro field puede ser el offset del campo, o el nombre del campo, o el nombre de la tabla un punto y el nombre del campo (nombre_tabla.nombre_campo). Si el nombre de la columna tiene un alias ('select foo as bar from...'), use el alias en vez del nombre de la columna.

Cuando trabaje con conjuntos de resultados grandes, debe considerar el uso de alguna de las funciones que cargan una fila entera (especificadas abajo). Ya que estas funciones devuelven el contenido de múltiples celdas en una única llamada, son MUCHO más rápidas que sybase_result().

Además, note que específicar un offset númerico en el parámetro field es mucho más rápido que específicar un nombre de campo o nombre_tabla.nombre_campo.

Alternativas recomendadas para alto rendimiento: **sybase_fetch_row()**, **sybase_fetch_array()**, **sybase_fetch_object()**.

sybase_select_db

(PHP 3, PHP 4 , PHP 5)

sybase_select_db -- seleccciona una base de datos Sybase

Descripción

int **sybase_select_db** (string database_name [, int link_identifier])

sybase_select_db() establece como activa la base de datos en el servidor asociada con el identificador de enlace específicado. Si no se específica un identificador de enlace, se asume el último enlace abierto. Si no hay un enlace abierto, la función intentará establecer un enlace como si **sybase_connect()** fuese llamada, y lo usará.

Cada llamada subsiguiente a **sybase_query()** será hecha en la base de datos activa.

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Cada llamada subsecuente a **sybase_query()** se hará en la base de datos activa.

Vea también **sybase_connect()**, **sybase_pconnect()**, **sybase_query()**

sybase_set_message_handler

(PHP 4 >= 4.3.0, PHP 5)

sybase_set_message_handler -- Fija el manejador a ser llamado cuando se emite un mensaje del sistema

Descripción

bool **sybase_set_message_handler** (callback handler [, int conexión])

sybase_set_message_handler() Fija una función definida por el usuario para manejar los mensajes generados por el servidor SYBASE. Se debe especificar el nombre de una función global, o usar una matriz para especificar una referencia a un objeto y al nombre de un método.

> **Nota:** Esta función se encuentra disponible solamente cuando se usan las bibliotecas CT de Sybase y no las bibliotecas DB.

El manejador debe recibir cinco argumentos en el siguiente orden: número del mensaje, severidad, estado, número de linea y descripción. Los primeros cuatro son de tipo entero. El último es una cadena. Si la función regresa **`FALSE`**, PHP genera un mensajer de error ordinario.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

> **Nota:** El parámetro *conexión* fue agregado en PHP 4.3.5.

Ejemplo 1. sybase_set_message_handler() llamada a una función

```
<?php
    function msg_handler($msgnumber, $severity, $state, $line, $text)
    {
        var_dump($msgnumber, $severity, $state, $line, $text);
    }

    sybase_set_message_handler('msg_handler');
?>
```

Ejemplo 2. sybase_set_message_handler() llamada a una clase

```
<?php
    class Sybase {
        function handler($msgnumber, $severity, $state, $line, $text)
        {
            var_dump($msgnumber, $severity, $state, $line, $text);
        }
    }

    $sybase= new Sybase();
    sybase_set_message_handler(array($sybase, 'handler'));
?>
```

Ejemplo 3. sybase_set_message_handler() Mensajes no manejados

```
<?php
    // Regresa FALSE, desde esta funcion para indicar que no se pudo
    // manejar el mensaje, El error es impreso como un WARNING, la forma
    // de usar esto, es si no hay un manejador de mensajes instalado.
    function msg_handler($msgnumber, $severity, $state, $line, $text)
    {
        if (257 == $msgnumber) {
            return false;
        }
        var_dump($msgnumber, $severity, $state, $line, $text);
    }

    sybase_set_message_handler('msg_handler');
?>
```

sybase_unbuffered_query

(PHP 4 >= 4.3.0, PHP 5)

sybase_unbuffered_query -- Envía una sentencia SQL a Sybase sin bloquear

Descripción

int **sybase_unbuffered_query** (string setencia, int identificador_de_enlace [, int almacena_resultado])

Devuelve un identificador de resultado Sybase positivo en caso exitoso, o **FALSE** ante un error.

> **Nota:** Esta función se encuentra disponible solamente cuando se usan las bibliotecas CT de Sybase y no las bibliotecas DB.

sybase_unbuffered_query() envía una consulta a la actual base de datos activa en el servidor que

está asociada con el identificador de enlace especificado. Si no se especifica un identificador de enlace, se asume el último enlace abierto. Si no hay un enlace abierto, la función intentará establecer un enlace como si **sybase_connect()** fuese llamada, y lo usará.

A diferencia de **sybase_query()**, **sybase_unbuffered_query()** leerá sólo la primera fila del resultado asociado. **sybase_fetch_array()** una función similar leerá mas filas según necesite. **sybase_data_seek()** lee hasta la fila solicitada. Este comportamiento puede producir un mejor desempeño en resultados asociados muy grandes.

sybase_num_rows() Sólo regresará el número de filas del resultado asociado que ha sido leido. Para Sybase, el número de filas en un resultado es desconocido, por lo tanto debe ser calculado por una implementación del cliente.

> **Nota:** Si no ha utilizado todos los resultados asociados antes de ejecutar una nueva sentencia, PHP generará una advertencia y cancelará todos los resultados pendientes. Para evitar esto, use **sybase_free_result()** el cual cancelará los resultados pendientes de una sentencia sin almacenamiento intermedio.

El parámetro opcional *almacena_resultado* puede estar en **FALSE** para indicar que el resultado asociado no debe ser traido a la memoria, y así minimiza el uso de la memoria, lo cuál es particularmente interesante con gran cantidad de filas en un resultado asociado.

Vea también **sybase_query()**

Ejemplo 1. Ejemplo sybase_unbuffered_query()

```
<?php
        $dbh= sybase_connect('SYBASE', '', '');
        $q= sybase_unbuffered_query('select firstname, lastname from huge_table', $dbh,
        sybase_data_seek($q, 10000);
        $i= 0;
        while ($row= sybase_fetch_row($q)) {
                echo $row[0] . ' ' . $row[0];
                if ($i++ > 40000) break;
        }
        sybase_free_result($q);
        sybase_close($dbh);
?>
```

CXXIV. TCP Wrappers Functions

Introducción

The TCP wrappers provides a classical unix mechanism which has been designed to check if the remote client is able to connect from the given IP address.

Instalación

Tcpwrap is currently available through PECL http://pecl.php.net/package/tcpwrap.

If PEAR is available on your *nix-like system you can use the pear installer to install the tcpwrap extension, by the following command: **pear -v install tcpwrap**.

You can always download the tar.gz package and install tcpwrap by hand:

Ejemplo 1. tcpwrap install by hand

```
gunzip tcpwrap-xxx.tgz
tar -xvf tcpwrap-xxx.tar
cd tcpwrap-xxx
phpize
./configure && make && make install
```

Tabla de contenidos

tcpwrap_check

(no version information, might be only in CVS)

tcpwrap_check -- Performs a tcpwrap check

Descripción

bool **tcpwrap_check** (string daemon, string address [, string user [, bool nodns]])

This function consults the `/etc/hosts.allow` and `/etc/hosts.deny` files to check if access to service *daemon* should be granted or denied for a client.

Lista de parámetros

daemon

The service name.

address

The client remote address. Can be either an IP address or a domain name.

user

An optional user name.

nodns

If *address* looks like domain name then DNS is used to resolve it to IP address; set *nodns* to **TRUE** to avoid this.

Valores retornados

This function returns **TRUE** if access should be granted, **FALSE** otherwise.

Ejemplos

Ejemplo 1. Deny all connections from localhost

If your /etc/hosts.deny file contains:

```
php: 127.0.0.1
```

And your code looks like:

```
<?php
if (!tcpwrap_check('php', $_SERVER['REMOTE_ADDR'])) {
  die('You are not welcome here');
}
?>
```

Ver también

For more details please consult hosts_access(3) man page.

CXXV. Tidy Functions

Introducción

Tidy is a binding for the Tidy HTML clean and repair utility which allows you to not only clean and otherwise manipulate HTML documents, but also traverse the document tree.

Requirimientos

To use Tidy, you will need libtidy installed, available on the tidy homepage http://tidy.sourceforge.net/.

Instalación

Tidy is currently available for PHP 4.3.x and PHP 5 as a PECL extension from http://pecl.php.net/package/tidy.

> **Nota:** Tidy 1.0 is just for PHP 4.3.x, while Tidy 2.0 is just for PHP 5.

If PEAR is available on your *nix-like system you can use the pear installer to install the tidy extension, by the following command: **pear -v install tidy**.

You can always download the tar.gz package and install tidy by hand:
Ejemplo 1. tidy install by hand in PHP 4.3.x

```
gunzip tidy-xxx.tgz
tar -xvf tidy-xxx.tar
cd tidy-xxx
phpize
./configure && make && make install
```

Windows users can download the extension dll `php_tidy.dll` from http://snaps.php.net/win32/PECL_STABLE/.

In PHP 5 you need only to compile using the *--with-tidy* option.

Configuración en tiempo de ejecución

El comportamiento de estas funciones está afectado por los valores definidos en `php.ini`.

Tabla 1. Tidy Configuration Options

Name	Default	Changeable
tidy.default_config	""	PHP_INI_SYSTEM
tidy.clean_output	0	PHP_INI_PERDIR

For further details and definitions of the PHP_INI_* constants, see the Apéndice H.

A continuación se presenta una corta explicación de las directivas de configuración.

tidy.default_config **string**

Default path for tidy config file.

tidy.clean_output **boolean**

Turns on/off the output repairing by Tidy.

Aviso
Do not turn on *tidy.clean_output* if you are generating non-html content such as dynamic images.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

Each **`TIDY_TAG_XXX`** represents a HTML tag. For example, **`TIDY_TAG_A`** represents a link tag. Each **`TIDY_ATTR_XXX`** represents a HTML atribute. For example **`TIDY_ATTR_HREF`** would represent the href atribute in the previous example.

The following constants are defined:

Tabla 2. tidy tag constants

constant
TIDY_TAG_UNKNOWN
TIDY_TAG_A

constant
TIDY_TAG_ABBR
TIDY_TAG_ACRONYM
TIDY_TAG_ALIGN
TIDY_TAG_APPLET
TIDY_TAG_AREA
TIDY_TAG_B
TIDY_TAG_BASE
TIDY_TAG_BASEFONT
TIDY_TAG_BDO
TIDY_TAG_BGSOUND
TIDY_TAG_BIG
TIDY_TAG_BLINK
TIDY_TAG_BLOCKQUOTE
TIDY_TAG_BODY
TIDY_TAG_BR
TIDY_TAG_BUTTON
TIDY_TAG_CAPTION
TIDY_TAG_CENTER
TIDY_TAG_CITE
TIDY_TAG_CODE
TIDY_TAG_COL
TIDY_TAG_COLGROUP
TIDY_TAG_COMMENT
TIDY_TAG_DD
TIDY_TAG_DEL
TIDY_TAG_DFN
TIDY_TAG_DIR
TIDY_TAG_DIV
TIDY_TAG_DL
TIDY_TAG_DT
TIDY_TAG_EM
TIDY_TAG_EMBED
TIDY_TAG_FIELDSET
TIDY_TAG_FONT
TIDY_TAG_FORM
TIDY_TAG_FRAME
TIDY_TAG_FRAMESET

constant
TIDY_TAG_H1
TIDY_TAG_H2
TIDY_TAG_H3
TIDY_TAG_H4
TIDY_TAG_H5
TIDY_TAG_H6
TIDY_TAG_HEAD
TIDY_TAG_HR
TIDY_TAG_HTML
TIDY_TAG_I
TIDY_TAG_IFRAME
TIDY_TAG_ILAYER
TIDY_TAG_IMG
TIDY_TAG_INPUT
TIDY_TAG_INS
TIDY_TAG_ISINDEX
TIDY_TAG_KBD
TIDY_TAG_KEYGEN
TIDY_TAG_LABEL
TIDY_TAG_LAYER
TIDY_TAG_LEGEND
TIDY_TAG_LI
TIDY_TAG_LINK
TIDY_TAG_LISTING
TIDY_TAG_MAP
TIDY_TAG_MARQUEE
TIDY_TAG_MENU
TIDY_TAG_META
TIDY_TAG_MULTICOL
TIDY_TAG_NOBR
TIDY_TAG_NOEMBED
TIDY_TAG_NOFRAMES
TIDY_TAG_NOLAYER
TIDY_TAG_NOSAFE
TIDY_TAG_NOSCRIPT
TIDY_TAG_OBJECT
TIDY_TAG_OL
TIDY_TAG_OPTGROUP

constant
TIDY_TAG_OPTION
TIDY_TAG_P
TIDY_TAG_PARAM
TIDY_TAG_PLAINTEXT
TIDY_TAG_PRE
TIDY_TAG_Q
TIDY_TAG_RP
TIDY_TAG_RT
TIDY_TAG_RTC
TIDY_TAG_RUBY
TIDY_TAG_S
TIDY_TAG_SAMP
TIDY_TAG_SCRIPT
TIDY_TAG_SELECT
TIDY_TAG_SERVER
TIDY_TAG_SERVLET
TIDY_TAG_SMALL
TIDY_TAG_SPACER
TIDY_TAG_SPAN
TIDY_TAG_STRIKE
TIDY_TAG_STRONG
TIDY_TAG_STYLE
TIDY_TAG_SUB
TIDY_TAG_TABLE
TIDY_TAG_TBODY
TIDY_TAG_TD
TIDY_TAG_TEXTAREA
TIDY_TAG_TFOOT
TIDY_TAG_TH
TIDY_TAG_THEAD
TIDY_TAG_TITLE
TIDY_TAG_TR
TIDY_TAG_TR
TIDY_TAG_TT
TIDY_TAG_U
TIDY_TAG_UL
TIDY_TAG_VAR
TIDY_TAG_WBR

constant
TIDY_TAG_XMP

Tabla 3. tidy attribute constants

constant
TIDY_ATTR_UNKNOWN
TIDY_ATTR_ABBR
TIDY_ATTR_ACCEPT
TIDY_ATTR_ACCEPT_CHARSET
TIDY_ATTR_ACCESSKEY
TIDY_ATTR_ACTION
TIDY_ATTR_ADD_DATE
TIDY_ATTR_ALIGN
TIDY_ATTR_ALINK
TIDY_ATTR_ALT
TIDY_ATTR_ARCHIVE
TIDY_ATTR_AXIS
TIDY_ATTR_BACKGROUND
TIDY_ATTR_BGCOLOR
TIDY_ATTR_BGPROPERTIES
TIDY_ATTR_BORDER
TIDY_ATTR_BORDERCOLOR
TIDY_ATTR_BOTTOMMARGIN
TIDY_ATTR_CELLPADDING
TIDY_ATTR_CELLSPACING
TIDY_ATTR_CHAR
TIDY_ATTR_CHAROFF
TIDY_ATTR_CHARSET
TIDY_ATTR_CHECKED
TIDY_ATTR_CITE
TIDY_ATTR_CLASS
TIDY_ATTR_CLASSID
TIDY_ATTR_CLEAR
TIDY_ATTR_CODE
TIDY_ATTR_CODEBASE
TIDY_ATTR_CODETYPE
TIDY_ATTR_COLOR
TIDY_ATTR_COLS
TIDY_ATTR_COLSPAN

constant
TIDY_ATTR_COMPACT
TIDY_ATTR_CONTENT
TIDY_ATTR_COORDS
TIDY_ATTR_DATA
TIDY_ATTR_DATAFLD
TIDY_ATTR_DATAPAGESIZE
TIDY_ATTR_DATASRC
TIDY_ATTR_DATETIME
TIDY_ATTR_DECLARE
TIDY_ATTR_DEFER
TIDY_ATTR_DIR
TIDY_ATTR_DISABLED
TIDY_ATTR_ENCODING
TIDY_ATTR_ENCTYPE
TIDY_ATTR_FACE
TIDY_ATTR_FOR
TIDY_ATTR_FRAME
TIDY_ATTR_FRAMEBORDER
TIDY_ATTR_FRAMESPACING
TIDY_ATTR_GRIDX
TIDY_ATTR_GRIDY
TIDY_ATTR_HEADERS
TIDY_ATTR_HEIGHT
TIDY_ATTR_HREF
TIDY_ATTR_HREFLANG
TIDY_ATTR_HSPACE
TIDY_ATTR_HTTP_ñ
TIDY_ATTR_ID
TIDY_ATTR_ISMAP
TIDY_ATTR_LABEL
TIDY_ATTR_LANG
TIDY_ATTR_LANGUAGE
TIDY_ATTR_LAST_MODIFIED
TIDY_ATTR_LAST_VISIT
TIDY_ATTR_LEFTMARGIN
TIDY_ATTR_LINK
TIDY_ATTR_LONGDESC
TIDY_ATTR_LOWSRC

constant
TIDY_ATTR_MARGINHEIGHT
TIDY_ATTR_MARGINWIDTH
TIDY_ATTR_MAXLENGTH
TIDY_ATTR_MEDIA
TIDY_ATTR_METHOD
TIDY_ATTR_MULTIPLE
TIDY_ATTR_NAME
TIDY_ATTR_NOHREF
TIDY_ATTR_NORESIZE
TIDY_ATTR_NOSHADE
TIDY_ATTR_NOWRAP
TIDY_ATTR_OBJECT
TIDY_ATTR_OnAFTERUPDATE
TIDY_ATTR_OnBEFOREUNLOAD
TIDY_ATTR_OnBEFOREUPDATE
TIDY_ATTR_OnBLUR
TIDY_ATTR_OnCHANGE
TIDY_ATTR_OnCLICK
TIDY_ATTR_OnDATAAVAILABLE
TIDY_ATTR_OnDATASETCHANGED
TIDY_ATTR_OnDATASETCOMPLETE
TIDY_ATTR_OnDBLCLICK
TIDY_ATTR_OnERRORUPDATE
TIDY_ATTR_OnFOCUS
TIDY_ATTR_OnKEYDOWN
TIDY_ATTR_OnKEYPRESS
TIDY_ATTR_OnKEYUP
TIDY_ATTR_OnLOAD
TIDY_ATTR_OnMOUSEDOWN
TIDY_ATTR_OnMOUSEMOVE
TIDY_ATTR_OnMOUSEOUT
TIDY_ATTR_OnMOUSEOVER
TIDY_ATTR_OnMOUSEUP
TIDY_ATTR_OnRESET
TIDY_ATTR_OnROWENTER
TIDY_ATTR_OnROWEXIT
TIDY_ATTR_OnSELECT

constant
TIDY_ATTR_OnSUBMIT
TIDY_ATTR_OnUNLOAD
TIDY_ATTR_PROFILE
TIDY_ATTR_PROMPT
TIDY_ATTR_RBSPAN
TIDY_ATTR_READONLY
TIDY_ATTR_REL
TIDY_ATTR_REV
TIDY_ATTR_RIGHTMARGIN
TIDY_ATTR_ROWS
TIDY_ATTR_ROWSPAN
TIDY_ATTR_RULES
TIDY_ATTR_SCHEME
TIDY_ATTR_SCOPE
TIDY_ATTR_SCROLLING
TIDY_ATTR_SELECTED
TIDY_ATTR_SHAPE
TIDY_ATTR_SHOWGRID
TIDY_ATTR_SHOWGRIDX
TIDY_ATTR_SHOWGRIDY
TIDY_ATTR_SIZE
TIDY_ATTR_SPAN
TIDY_ATTR_SRC
TIDY_ATTR_STANDBY
TIDY_ATTR_START
TIDY_ATTR_STYLE
TIDY_ATTR_SUMMARY
TIDY_ATTR_TABINDEX
TIDY_ATTR_TARGET
TIDY_ATTR_TEXT
TIDY_ATTR_TITLE
TIDY_ATTR_TOPMARGIN
TIDY_ATTR_TYPE
TIDY_ATTR_USEMAP
TIDY_ATTR_VALIGN
TIDY_ATTR_VALUE
TIDY_ATTR_VALUETYPE
TIDY_ATTR_VERSION

constant
TIDY_ATTR_VLINK
TIDY_ATTR_VSPACE
TIDY_ATTR_WIDTH
TIDY_ATTR_WRAP
TIDY_ATTR_XML_LANG
TIDY_ATTR_XML_SPACE
TIDY_ATTR_XMLNS

Tabla 4. tidy nodetype constants

constant	description
TIDY_NODETYPE_ROOT	root node
TIDY_NODETYPE_DOCTYPE	doctype
TIDY_NODETYPE_COMMENT	HTML comment
TIDY_NODETYPE_PROCINS	Processing Instruction
TIDY_NODETYPE_TEXT	Text
TIDY_NODETYPE_START	start tag
TIDY_NODETYPE_END	end tag
TIDY_NODETYPE_STARTEND	empty tag
TIDY_NODETYPE_CDATA	CDATA
TIDY_NODETYPE_SECTION	XML section
TIDY_NODETYPE_ASP	ASP code
TIDY_NODETYPE_JSTE	JSTE code
TIDY_NODETYPE_PHP	PHP code
TIDY_NODETYPE_XMLDECL	XML declaration

Ejemplos

This simple example shows basic Tidy usage.

Ejemplo 2. Basic Tidy usage

```
<?php
ob_start();
?>
<html>a html document</html>
<?
$html = ob_get_clean();

// Specify configuration
$config = array(
           'indent'         => true,
           'output-xhtml'   => true,
           'wrap'           => 200);

// Tidy
$tidy = new tidy;
$tidy->parseString($html, $config, 'utf8');
$tidy->cleanRepair();

// Output
echo $tidy;
?>
```

Tabla de contenidos

tidy_parse_string -- Parse a document stored in a string
tidy_repair_file -- Repair a file and return it as a string
tidy_repair_string -- Repair a string using an optionally provided configuration file
tidy_reset_config -- Restore Tidy configuration to default values
tidy_save_config -- Save current settings to named file
tidy_set_encoding -- Set the input/output character encoding for parsing markup
tidy_setopt -- Updates the configuration settings for the specified tidy document
tidy_warning_count -- Returns the Number of Tidy warnings encountered for specified document
tidyNode->isAsp -- Returns true if this node is ASP
tidyNode->isPhp -- Returns true if this node is PHP

ob_tidyhandler

(PHP 5)

ob_tidyhandler -- ob_start callback function to repair the buffer

Description

string **ob_tidyhandler** (string input [, int mode])

ob_tidyhandler() is intended to be used as a callback function for **ob_start()** to repair the buffer.

Ejemplo 1. ob_tidyhandler() example

```
<?php
ob_start('ob_tidyhandler');

echo '<p>test</i>';
?>
```

El resultado del ejemplo seria:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html>
<head>
<title></title>
</head>
<body>
<p>test</p>
</body>
</html>
```

See also **ob_start()**.

tidy_access_count

(PHP 5)

tidy_access_count -- Returns the Number of Tidy accessibility warnings encountered for specified document

Description

int **tidy_access_count** (tidy object)

tidy_access_count() returns the number of accessibility warnings found for the specified document.

Nota: Due to the design of the TidyLib, you must call **tidy_diagnose()** before **tidy_access_count()** or it will return always *0*. You must also need to enable the *accessibility-check* option.

Ejemplo 1. tidy_access_count() example

```
<?php

$html ='<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html><head><title>Title</title></head>
<body>

<p><img src="img.png"></p>

</body></html>';


// select the accessibility check level: 1, 2 or 3
$config = array('accessibility-check' => 3);

$tidy = new tidy();
$tidy->parseString($html, $config);
$tidy->CleanRepair();

/* Never forget to call this! */
$tidy->diagnose();

echo tidy_access_count($tidy); //5

?>
```

See also **tidy_error_count()** and **tidy_warning_count()**.

tidy_clean_repair

(PHP 5)

tidy_clean_repair -- Execute configured cleanup and repair operations on parsed markup

Description

Procedural style:

bool **tidy_clean_repair** (tidy object)

Object oriented style:

bool **tidy->cleanRepair** (void)

This function cleans and repairs the given tidy *object*.

Ejemplo 1. tidy_clean_repair() example

```
<?php
$html = '<p>test</I>';

$tidy = tidy_parse_string($html);
tidy_clean_repair($tidy);

echo $tidy;
?>
```

El resultado del ejemplo seria:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html>
<head>
<title></title>
</head>
<body>
<p>test</p>
</body>
</html>
```

See also **tidy_repair_file()** and **tidy_repair_string()**.

tidy_config_count

(PHP 5)

tidy_config_count -- Returns the Number of Tidy configuration errors encountered for specified document

Description

int **tidy_config_count** (tidy object)

tidy_config_count() returns the number of errors encountered in the configuration of the specified tidy *object*.

Ejemplo 1. tidy_config_count() example

```
<?php
$html = '<p>test</I>';

$config = array('doctype' => 'bogus');

$tidy = tidy_parse_string($html, $config);

/* This outputs 1, because 'bogus' isn't a valid doctype */
echo tidy_config_count($tidy);
?>
```

tidy::__construct

(no version information, might be only in CVS)

tidy::__construct -- Constructs a new tidy object

Description

tidy **tidy::__construct** ([string filename [, mixed config [, string encoding [, bool use_include_path]]]])

tidy::__construct() constructs a new tidy object.

If the *filename* parameter is given, this function will also read that file and initialize the object with the file, acting like **tidy_parse_file()**.

El parametro *config* puede ser enviado como una matriz o como una cadena alfanumerica. Si se

envia como una cadena alfanumerica, signica que sera interpretado como el nombre del fichero de configuracion, de lo contrario, como opciones. Consultar http://tidy.sourceforge.net/docs/quickref.html para una explicacion de todas las opciones.

El parametro *encoding* define la codificacion para las salidas/entradas de los documentos. Los valores que se pueden utilizar son ascii, latin0, latin1, raw, utf8, iso2022, mac, win1252, ibm858, utf16, utf16le, utf16be, big5 and shiftjis.

Ejemplo 1. tidy::__construct() example

```
<?php

$html = <<< HTML

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head><title>title</title></head>
<body>
<p>paragraph <bt />
text</p>
</body></html>

HTML;

$tidy = new tidy;
$tidy->parseString($html);

$tidy->CleanRepair();

if ($tidy->errorBuffer) {
    echo "The following errors were detected:\n";
    echo $tidy->errorBuffer;
}

?>
```

El resultado del ejemplo seria:

```
The following errors were detected:
line 8 column 14 - Error: <bt> is not recognized!
line 8 column 14 - Warning: discarding unexpected <bt>
```

See also **tidy_parse_file()** and **tidy_parse_string()**.

tidy_diagnose

(PHP 5)

tidy_diagnose -- Run configured diagnostics on parsed and repaired markup

Description

Procedural style:

bool **tidy_diagnose** (tidy object)

Object oriented style:

bool **tidy->diagnose** (void)

tidy_diagnose() runs diagnostic tests on the given tidy *object*, adding some more information about the document in the error buffer.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplo 1. tidy_diagnose() example

```
<?php

$html = <<< HTML
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<p>paragraph</p>
HTML;

$tidy = tidy_parse_string($html);
$tidy->CleanRepair();

// note the difference between the two outputs
echo tidy_get_error_buffer($tidy) . "\n";

$tidy->diagnose();
echo tidy_get_error_buffer($tidy);

?>
```

El resultado del ejemplo seria:

```
line 5 column 1 - Warning: <p> isn't allowed in <head> elements
line 5 column 1 - Warning: inserting missing 'title' element

line 5 column 1 - Warning: <p> isn't allowed in <head> elements
line 5 column 1 - Warning: inserting missing 'title' element
Info: Doctype given is "-//W3C//DTD XHTML 1.0 Strict//EN"
Info: Document content looks like XHTML 1.0 Strict
2 warnings, 0 errors were found!
```

See also **tidy_get_error_buffer()**.

tidy_error_count

(PHP 5)

tidy_error_count -- Returns the Number of Tidy errors encountered for specified document

Description

int **tidy_error_count** (tidy object)

tidy_error_count() returns the number of Tidy errors encountered for the specified document.

Ejemplo 1. tidy_error_count() example

```
<?php
$html = '<p>test</i>
<bogustag>bogus</bogustag>';

$tidy = tidy_parse_string($html);

echo tidy_error_count($tidy) . "\n"; //1

echo $tidy->ErrorBuffer;
?>
```

The above example will output:

```
1
line 1 column 1 - Warning: missing <!DOCTYPE> declaration
line 1 column 8 - Warning: discarding unexpected </i>
line 2 column 1 - Error: <bogustag> is not recognized!
line 2 column 1 - Warning: discarding unexpected <bogustag>
line 2 column 16 - Warning: discarding unexpected </bogustag>
line 1 column 1 - Warning: inserting missing 'title' element
```

See also **tidy_access_count()** and **tidy_warning_count()**.

tidy_get_body

(PHP 5)

tidy_get_body -- Returns a tidyNode Object starting from the <body> tag of the tidy parse tree

Description

Procedural style:

tidyNode **tidy_get_body** (tidy object)

Object oriented style:

tidyNode **tidy->body** (void)

This function returns a tidyNode object starting from the <body> tag of the tidy parse tree.

Ejemplo 1. tidy_get_body() example

```
<?php
$html = '
<html>
  <head>
    <title>test</title>
  </head>
  <body>
    <p>paragraph</p>
  </body>
</html>';

$tidy = tidy_parse_string($html);

$body = tidy_get_body($tidy);
echo $body->value;
?>
```

El resultado del ejemplo seria:

```
<body>
<p>paragraph</p>
</body>
```

> **Nota:** Esta funcion esta disponible solamente con Zend Engine 2, o lo que es lo mismo PHP >= 5.0.0.

See also **tidy_get_head()** and **tidy_get_html()**.

tidy_get_config

(PHP 5)

tidy_get_config -- Get current Tidy configuration

Description

Procedural style:

array **tidy_get_config** (tidy object)

Object oriented style:

array **tidy->getConfig** (void)

tidy_get_config() returns an array with the configuration options in use by the given tidy *object*.

For an explanation about each option, visit http://tidy.sourceforge.net/docs/quickref.html.

Ejemplo 1. tidy_get_config() example

```
<?php
$html = '<p>test</p>';
$config = array('indent' => TRUE,
                'output-xhtml' => TRUE,
                'wrap' => 200);

$tidy = tidy_parse_string($html, $config);

print_r(tidy_get_config($tidy));
?>
```

El resultado del ejemplo seria:

```
Array
(
    [indent-spaces] => 2
    [wrap] => 200
    [tab-size] => 8
    [char-encoding] => 1
    [input-encoding] => 3
    [output-encoding] => 1
    [newline] => 1
    [doctype-mode] => 1
    [doctype] =>
    [repeated-attributes] => 1
    [alt-text] =>
    [slide-style] =>
    [error-file] =>
    [output-file] =>
    [write-back] =>
    [markup] => 1
    [show-warnings] => 1
    [quiet] =>
    [indent] => 1
    [hide-endtags] =>
    [input-xml] =>
    [output-xml] => 1
    [output-xhtml] => 1
    [output-html] =>
    [add-xml-decl] =>
    [uppercase-tags] =>
    [uppercase-attributes] =>
    [bare] =>
    [clean] =>
    [logical-emphasis] =>
    [drop-proprietary-attributes] =>
    [drop-font-tags] =>
    [drop-empty-paras] => 1
    [fix-bad-comments] => 1
    [break-before-br] =>
    [split] =>
    [numeric-entities] =>
    [quote-marks] =>
    [quote-nbsp] => 1
    [quote-ampersand] => 1
    [wrap-attributes] =>
    [wrap-script-literals] =>
    [wrap-sections] => 1
    [wrap-asp] => 1
    [wrap-jste] => 1
    [wrap-php] => 1
    [fix-backslash] => 1
    [indent-attributes] =>
    [assume-xml-procins] =>
    [add-xml-space] =>
    [enclose-text] =>
    [enclose-block-text] =>
    [keep-time] =>
    [word-2000] =>
    [tidy-mark] =>
    [gnu-emacs] =>
    [gnu-emacs-file] =>
    [literal-attributes] =>
    [show-body-only] =>
    [fix-uri] => 1
    [lower-literals] => 1
    [hide-comments] =>
    [indent-cdata] =>
    [force-output] => 1
    [show-errors] => 6
    [ascii-chars] => 1
    [join-classes] =>
    [join-styles] => 1
    [escape-cdata] =>
    [language] =>
```

See also **tidy_reset_config()** and **tidy_save_config()**.

tidy_get_error_buffer

(PHP 5)

tidy_get_error_buffer -- Return warnings and errors which occurred parsing the specified document

Description

Procedural style:

string **tidy_get_error_buffer** (tidy object)

Object oriented style (property):

class **tidy** {

string errorBuffer

}

tidy_get_error_buffer() returns warnings and errors which occurred parsing the specified document.

Ejemplo 1. tidy_get_error_buffer() example

```
<?php
$html = '<p>paragraph</p>';

$tidy = tidy_parse_string($html);

echo tidy_get_error_buffer($tidy);
/* or in OO: */
echo $tidy->errorBuffer;
?>
```

El resultado del ejemplo seria:

```
line 1 column 1 - Warning: missing <!DOCTYPE> declaration
line 1 column 1 - Warning: inserting missing 'title' element
```

See also **tidy_access_count()**, **tidy_error_count()** and **tidy_warning_count()**.

tidy_get_head

(PHP 5)

tidy_get_head -- Returns a tidyNode Object starting from the <head> tag of the tidy parse tree

Description

Procedural style:

tidyNode **tidy_get_head** (tidy object)

Object oriented style:

tidyNode **tidy->head** (void)

This function returns a tidyNode object starting from the <head> tag of the tidy parse tree.

Ejemplo 1. tidy_get_head() example

```
<?php
$html = '
<html>
  <head>
    <title>test</title>
  </head>
  <body>
    <p>paragraph</p>
  </body>
</html>';

$tidy = tidy_parse_string($html);

$head = tidy_get_head($tidy);
echo $head->value;
?>
```

El resultado del ejemplo seria:

```
<head>
<title>test</title>
</head>
```

> **Nota:** Esta funcion esta disponible solamente con Zend Engine 2, o lo que es lo mismo PHP >= 5.0.0.

See also **tidy_get_body()** and **tidy_get_html()**.

tidy_get_html_ver

(PHP 5)

tidy_get_html_ver -- Get the Detected HTML version for the specified document

Description

Procedural style:

int **tidy_get_html_ver** (tidy object)

Object oriented style:

int **tidy->getHtmlVer** (void)

tidy_get_html_ver() returns the detected HTML version for the specified tidy *object*.

Aviso
This function is not yet implemented in the Tidylib itself, so it always return *0*.

tidy_get_html

(PHP 5)

tidy_get_html -- Returns a tidyNode Object starting from the <html> tag of the tidy parse tree

Description

Procedural style:

tidyNode **tidy_get_html** (tidy object)

Object oriented style:

tidyNode **tidy->html** (void)

This function returns a tidyNode object starting from the <html> tag of the tidy parse tree.

Ejemplo 1. tidy_get_html() example

```
<?php
$html = '
<html>
  <head>
    <title>test</title>
  </head>
  <body>
    <p>paragraph</p>
  </body>
</html>';

$tidy = tidy_parse_string($html);

$html = tidy_get_html($tidy);
echo $html->value;
?>
```

El resultado del ejemplo seria:

```
<html>
<head>
<title>test</title>
</head>
<body>
<p>paragraph</p>
</body>
</html>
```

> **Nota:** Esta funcion esta disponible solamente con Zend Engine 2, o lo que es lo mismo PHP >= 5.0.0.

See also **tidy_get_body()** and **tidy_get_head()**.

tidy_get_output

(PHP 5)

tidy_get_output -- Return a string representing the parsed tidy markup

Description

string **tidy_get_output** (tidy object)

tidy_get_output() returns a string with the repaired html.

Ejemplo 1. tidy_get_output() example

```
<?php

$html = '<p>paragraph</i>';
$tidy = tidy_parse_string($html);

$tidy->CleanRepair();

echo tidy_get_output($tidy);
?>
```

El resultado del ejemplo seria:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html>
<head>
<title></title>
</head>
<body>
<p>paragraph</p>
</body>
</html>
```

tidy_get_release

(PHP 5)

tidy_get_release -- Get release date (version) for Tidy library

Description

Procedural style:

string **tidy_get_release** (void)

Object oriented style:

string **tidy->getRelease** (void)

This function returns a string with the release date of the Tidy library.

tidy_get_root

(PHP 5)

tidy_get_root -- Returns a tidyNode object representing the root of the tidy parse tree

Description

Procedural style:

tidyNode **tidy_get_root** (tidy object)

Object oriented style:

tidyNode **tidy->root** (void)

Returns a tidyNode object representing the root of the tidy parse tree.

Ejemplo 1. dump nodes

```
<?php

$html = <<< HTML
<html><body>

<p>paragraph</p>
<br/>

</body></html>
HTML;

$tidy = tidy_parse_string($html);
dump_nodes($tidy->root(), 1);


function dump_nodes($node, $indent) {

    if($node->hasChildren()) {
        foreach($node->child as $child) {
            echo str_repeat('.', $indent*2) . ($child->name ? $child->name : '"'.$child

            dump_nodes($child, $indent+1);
        }
    }
}
?>
```

El resultado del ejemplo seria:

```
..html
....head
......title
....body
......p
........"paragraph"
......br
```

> **Nota:** Esta funcion esta disponible solamente con Zend Engine 2, o lo que es lo mismo PHP >= 5.0.0.

tidy_get_status

(PHP 5)

tidy_get_status -- Get status of specified document

Description

Procedural style:

int **tidy_get_status** (tidy object)

Object oriented style:

int **tidy->getStatus** (void)

tidy_get_status() returns the status for the specified tidy *object*. It returns 0 if no error/warning was raised, 1 for warnings or accessibility errors, or 2 for errors.

Ejemplo 1. tidy_get_status() example

```
<?php
$html = '<p>paragraph</i>';
$tidy = tidy_parse_string($html);

$html2 = '<bogus>test</bogus>';
$tidy2 = tidy_parse_string($html2);

echo tidy_get_status($tidy); //1

echo tidy_get_status($tidy2); //2
?>
```

tidy_getopt

(PHP 5)

tidy_getopt -- Returns the value of the specified configuration option for the tidy document

Description

Procedural style:

mixed **tidy_getopt** (tidy object, string option)

Object oriented style:

mixed **tidy->getOpt** (string option)

tidy_getopt() returns the value of the specified *option* for the specified tidy *object*. The return type depends on the type of the specified *option*. You will find a list with each configuration option and their types at: http://tidy.sourceforge.net/docs/quickref.html.

Ejemplo 1. tidy_getopt() example

```
<?php

$html ='<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html><head><title>Title</title></head>
<body>

<p><img src="img.png"></p>

</body></html>';

$config = array('accessibility-check' => 3,
                'alt-text' => 'some text');

$tidy = new tidy();
$tidy->parseString($html, $config);


var_dump($tidy->getOpt('accessibility-check')); //integer
var_dump($tidy->getOpt('lower-literals')); //boolean
var_dump($tidy->getOpt('alt-text')); //string

?>
```

El resultado del ejemplo seria:

```
int(3)
bool(true)
string(9) "some text"
```

tidy_is_xhtml

(PHP 5)

tidy_is_xhtml -- Indicates if the document is a XHTML document

Description

Procedural style:

bool **tidy_is_xhtml** (tidy object)

Object oriented style:

bool **tidy->isXhtml** (void)

This function returns **TRUE** if the specified tidy *object* is a XHTML document, or **FALSE** otherwise.

Aviso
This function is not yet implemented in the Tidylib itself, so it always return **FALSE**.

tidy_is_xml

(PHP 5)

tidy_is_xml -- Indicates if the document is a generic (non HTML/XHTML) XML document

Description

Procedural style:

bool **tidy_is_xml** (tidy object)

Object oriented style:

bool **tidy->isXml** (void)

This function returns **TRUE** if the specified tidy *object* is a generic XML document (non HTML/XHTML), or **FALSE** otherwise.

Aviso
This function is not yet implemented in the Tidylib itself, so it always return **FALSE**.

tidy_load_config

(no version information, might be only in CVS)

tidy_load_config -- Load an ASCII Tidy configuration file with the specified encoding

Description

void **tidy_load_config** (string filename, string encoding)

This function loads a Tidy configuration file, with the specified *encoding*.

> **Nota:** Esta funcion esta disponible solamente con Tidy 1.0. Se dejo de utilizar en Tidy 2.0 y por lo tanto no se encuentra dispobible actualmente.

tidy_node->children

(no version information, might be only in CVS)

tidy_node->children -- Returns an array of child nodes

Description

array **tidy_node->children** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

tidy_node->get_attr

(no version information, might be only in CVS)

tidy_node->get_attr -- Return the attribute with the provided attribute id

Description

tidy_attr **tidy_node->get_attr** (int attrib_id)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

tidy_node->get_nodes

(no version information, might be only in CVS)

tidy_node->get_nodes -- Return an array of nodes under this node with the specified id

Description

array **tidy_node->get_nodes** (int node_id)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

tidy_node->hasChildren

(no version information, might be only in CVS)

tidy_node->hasChildren -- Returns true if this node has children

Description

bool **tidy_node->hasChildren** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

Nota: This function was named **tidy_node->has_children()** in PHP 4/Tidy 1.

tidy_node->hasSiblings

(no version information, might be only in CVS)

tidy_node->hasSiblings -- Returns true if this node has siblings

Description

bool **tidy_node->hasSiblings** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

Nota: This function was named **tidy_node->has_siblings()** in PHP 4/Tidy 1.

tidy_node->isComment

(no version information, might be only in CVS)

tidy_node->isComment -- Returns true if this node represents a comment

Description

bool **tidy_node->isComment** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

Nota: This function was named **tidy_node->is_comment()** in PHP 4/Tidy 1.

tidy_node->isHtml

(no version information, might be only in CVS)

tidy_node->isHtml -- Returns true if this node is part of a HTML document

Description

bool **tidy_node->isHtml** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

Nota: This function was named **tidy_node->is_html()** in PHP 4/Tidy 1.

tidy_node->isJste

(no version information, might be only in CVS)

tidy_node->isJste -- Returns true if this node is JSTE

Description

bool **tidy_node->isJste** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

Nota: This function was named **tidy_node->is_jste()** in PHP 4/Tidy 1.

tidy_node->isText

(no version information, might be only in CVS)

tidy_node->isText -- Returns true if this node represents text (no markup)

Description

bool **tidy_node->isText** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

Nota: This function was named **tidy_node->is_text()** in PHP 4/Tidy 1.

tidy_node->isXhtml

(no version information, might be only in CVS)

tidy_node->isXhtml -- Returns true if this node is part of a XHTML document

Description

bool **tidy_node->isXhtml** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

Nota: This functions was named **tidy_node->is_xhtml()** in PHP 4/Tidy 1.

tidy_node->isXml

(no version information, might be only in CVS)

tidy_node->isXml -- Returns true if this node is part of a XML document

Description

bool **tidy_node->isXml** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

Nota: This function was named **tidy_node->is_xml()** in PHP 4/Tidy 1.

tidy_node->next

(no version information, might be only in CVS)

tidy_node->next -- Returns the next sibling to this node

Description

tidy_node **tidy_node->next** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

tidy_node->prev

(no version information, might be only in CVS)

tidy_node->prev -- Returns the previous sibling to this node

Description

tidy_node **tidy_node->prev** (void)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

tidy_parse_file

(PHP 5)

tidy_parse_file -- Parse markup in file or URI

Description

Procedural style:

tidy **tidy_parse_file** (string filename [, mixed config [, string encoding [, bool use_include_path]]])

Object oriented style:

bool **tidy->parseFile** (string filename [, mixed config [, string encoding [, bool use_include_path]]])

This function parses the given file.

El parametro *config* puede ser enviado como una matriz o como una cadena alfanumerica. Si se envia como una cadena alfanumerica, signica que sera interpretado como el nombre del fichero de configuracion, de lo contrario, como opciones. Consultar http://tidy.sourceforge.net/docs/quickref.html para una explicacion de todas las opciones.

El parametro *encoding* define la codificacion para las salidas/entradas de los documentos. Los valores que se pueden utilizar son ascii, latin0, latin1, raw, utf8, iso2022, mac, win1252, ibm858, utf16, utf16le, utf16be, big5 and shiftjis.

Ejemplo 1. tidy_parse_file() example

```
<?php
$tidy = tidy_parse_file('file.html');

$tidy->cleanRepair();

if(!empty($tidy->errorBuffer)) {
    echo "The following errors or warnings occured:\n";
    echo $tidy->errorBuffer;
}
?>
```

> **Nota:** Los parametros adicionales *config* y *encoding* se encuentran disponibles a partir de Tidy 2.0.

See also **tidy_parse_string()**, **tidy_parse_string()** and **tidy_repair_string()**.

tidy_parse_string

(PHP 5)

tidy_parse_string -- Parse a document stored in a string

Description

Procedural style:

tidy **tidy_parse_string** (string input [, mixed config [, string encoding]])

Object oriented style:

bool **tidy->parseString** (string input [, mixed config [, string encoding]])

tidy_parse_string() parses a document stored in a string.

El parametro *config* puede ser enviado como una matriz o como una cadena alfanumerica. Si se

envia como una cadena alfanumerica, signica que sera interpretado como el nombre del fichero de configuracion, de lo contrario, como opciones. Consultar http://tidy.sourceforge.net/docs/quickref.html para una explicacion de todas las opciones.

El parametro *encoding* define la codificacion para las salidas/entradas de los documentos. Los valores que se pueden utilizar son ascii, latin0, latin1, raw, utf8, iso2022, mac, win1252, ibm858, utf16, utf16le, utf16be, big5 and shiftjis.

Ejemplo 1. tidy_parse_string() example

```
<?php
ob_start();
?>

<html>
  <head>
   <title>test</title>
  </head>
  <body>
   <p>error<br>another line</i>
  </body>
</html>

<?php

$buffer = ob_get_clean();
$config = array('indent' => TRUE,
                'output-xhtml' => TRUE,
                'wrap' => 200);

$tidy = tidy_parse_string($buffer, $config, 'UTF8');

$tidy->cleanRepair();
echo $tidy;
?>
```

The above example will output:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>
      test
    </title>
  </head>
  <body>
    <p>
      error<br />
      another line
    </p>
  </body>
</html>
```

> **Nota:** Los parametros adicionales *config* y *encoding* se encuentran disponibles a partir de Tidy 2.0.

See also **tidy_parse_file()**, **tidy_repair_file()** and **tidy_repair_string()**.

tidy_repair_file

(PHP 5)

tidy_repair_file -- Repair a file and return it as a string

Description

string **tidy_repair_file** (string filename [, mixed config [, string encoding [, bool use_include_path]]])

This function repairs the given file and returns it as a string.

El parametro *config* puede ser enviado como una matriz o como una cadena alfanumerica. Si se envia como una cadena alfanumerica, signica que sera interpretado como el nombre del fichero de configuracion, de lo contrario, como opciones. Consultar http://tidy.sourceforge.net/docs/quickref.html para una explicacion de todas las opciones.

El parametro *encoding* define la codificacion para las salidas/entradas de los documentos. Los valores que se pueden utilizar son ascii, latin0, latin1, raw, utf8, iso2022, mac, win1252, ibm858, utf16, utf16le, utf16be, big5 and shiftjis.

Ejemplo 1. tidy_repair_file() example

```
<?php
$file = 'file.html';

$repaired = tidy_repair_file($file);
rename($file, $file . '.bak');

file_put_contents($file, $repaired);
?>
```

> **Nota:** Los parametros adicionales *config* y *encoding* se encuentran disponibles a partir de Tidy 2.0.

See also **tidy_parse_file()**, **tidy_parse_string()** and **tidy_repair_string()**.

tidy_repair_string

(PHP 5)

tidy_repair_string -- Repair a string using an optionally provided configuration file

Description

string **tidy_repair_string** (string data [, mixed config [, string encoding]])

This function repairs the given string.

El parametro *config* puede ser enviado como una matriz o como una cadena alfanumerica. Si se envia como una cadena alfanumerica, signica que sera interpretado como el nombre del fichero de configuracion, de lo contrario, como opciones. Consultar http://tidy.sourceforge.net/docs/quickref.html para una explicacion de todas las opciones.

El parametro *encoding* define la codificacion para las salidas/entradas de los documentos. Los valores que se pueden utilizar son ascii, latin0, latin1, raw, utf8, iso2022, mac, win1252, ibm858, utf16, utf16le, utf16be, big5 and shiftjis.

Ejemplo 1. tidy_repair_string() example

```
<?php
ob_start();
?>

<html>
  <head>
    <title>test</title>
  </head>
  <body>
    <p>error</i>
  </body>
</html>

<?php

$buffer = ob_get_clean();
$tidy = tidy_repair_string($buffer);

echo $tidy;
?>
```

The above example will output:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html>
<head>
<title>test</title>
</head>
<body>
<p>error</p>
</body>
</html>
```

> **Nota:** Los parametros adicionales *config* y *encoding* se encuentran disponibles a partir de Tidy 2.0.

See also **tidy_parse_file()**, **tidy_parse_string()** and **tidy_repair_file()**.

tidy_reset_config

(no version information, might be only in CVS)

tidy_reset_config -- Restore Tidy configuration to default values

Description

bool **tidy_reset_config** (void)

This function restores the Tidy configuration to the default values.

> **Nota:** Esta funcion esta disponible solamente con Tidy 1.0. Se dejo de utilizar en Tidy 2.0 y por lo tanto no se encuentra dispobible actualmente.

tidy_save_config

(no version information, might be only in CVS)

tidy_save_config -- Save current settings to named file

Description

bool **tidy_save_config** (string filename)

tidy_save_config() saves current settings to the specified file. Only non-default values are written.

See also **tidy_get_config()**, **tidy_getopt()**, **tidy_reset_config()** and **tidy_setopt()**.

> **Nota:** Esta funcion esta disponible solamente con Tidy 1.0. Se dejo de utilizar en Tidy 2.0 y por lo tanto no se encuentra dispobible actualmente.

tidy_set_encoding

(no version information, might be only in CVS)

tidy_set_encoding -- Set the input/output character encoding for parsing markup

Description

bool **tidy_set_encoding** (string encoding)

Sets the encoding for input/output documents. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo. Possible values for *encoding* are ascii, latin0, latin1, raw, utf8, iso2022, mac, win1252, ibm858, utf16, utf16le, utf16be, big5 and shiftjis

> **Nota:** Esta funcion esta disponible solamente con Tidy 1.0. Se dejo de utilizar en Tidy 2.0 y por lo tanto no se encuentra dispobible actualmente.

tidy_setopt

(no version information, might be only in CVS)

tidy_setopt -- Updates the configuration settings for the specified tidy document

Description

bool **tidy_setopt** (string option, mixed value)

tidy_setopt() updates the specified *option* with a new *value*.

Ejemplo 1. tidy_setopt() example

```
<?php
$html = '<p>test</i>';

$tidy = tidy_parse_string($html);

tidy_setopt('indent', FALSE);
?>
```

See also **tidy_getopt()**, **tidy_get_config()**, **tidy_reset_config()** and **tidy_save_config()**.

> **Nota:** Esta funcion esta disponible solamente con Tidy 1.0. Se dejo de utilizar en Tidy

2.0 y por lo tanto no se encuentra dispobible actualmente.

tidy_warning_count

(PHP 5)

tidy_warning_count -- Returns the Number of Tidy warnings encountered for specified document

Description

int **tidy_warning_count** (tidy object)

tidy_warning_count() returns the number of Tidy warnings encountered for the specified document.

Ejemplo 1. tidy_warning_count() example

```
<?php
$html = '<p>test</i>
<bogustag>bogus</bogustag>';

$tidy = tidy_parse_string($html);

echo tidy_error_count($tidy) . "\n"; //1
echo tidy_warning_count($tidy) . "\n"; //5
?>
```

See also **tidy_access_count()** and **tidy_error_count()**.

tidyNode->isAsp

(no version information, might be only in CVS)

tidyNode->isAsp -- Returns true if this node is ASP

Description

bool **tidyNode->isAsp** (void)

This functions returns **TRUE** if the current node is ASP, or **FALSE** otherwise.

Nota: This function was named **tidy_node->is_asp()** in PHP 4/Tidy 1.

tidyNode->isPhp

(no version information, might be only in CVS)

tidyNode->isPhp -- Returns true if this node is PHP

Description

bool **tidyNode->isPhp** (void)

Returns **TRUE** if the current node is PHP code, **FALSE** otherwise.

Ejemplo 1. get the PHP code from a mixed HTML/PHP document

```
<?php

$html = <<< HTML
<html><head>
<?php echo '<title>title</title>'; ?>
</head>
<body>

<?php
echo 'hello world!';
?>

</body></html>
HTML;


$tidy = tidy_parse_string($html);
$num = 0;

get_php($tidy->html());


function get_php($node) {

    // check if the current node is PHP code
    if($node->isPhp()) {
        echo "\n\n# PHP node #" . ++$GLOBALS['num'] . "\n";
        echo $node->value;
    }

    // check if the current node has childrens
    if($node->hasChildren()) {
        foreach($node->child as $child) {
            get_php($child);
        }
    }
}

?>
```

El resultado del ejemplo seria:

```
# PHP node #1
<?php echo '<title>title</title>'; ?>

# PHP node #2
<?php
echo 'hello world!';
?>
```

Nota: This function was named **tidy_node->is_php()** in PHP 4/Tidy 1.

CXXVI. Tokenizer Functions

Introducción

The tokenizer functions provide an interface to the PHP tokenizer embedded in the Zend Engine. Using these functions you may write your own PHP source analyzing or modification tools without having to deal with the language specification at the lexical level.

See also the appendix about tokens.

Requirimientos

No se necesitan bibliotecas externas para construir esta extensión

---

Instalación

Beginning with PHP 4.3.0 these functions are enabled by default. For older versions you have to configure and compile PHP with *--enable-tokenizer*. You can disable tokenizer support with *--disable-tokenizer*.

La versión para Windows de *PHP* tiene soporte nativo para esta extensión. No se necesita cargar ninguna extensión adicional para usar estas funciones.

> **Nota:** Builtin support for tokenizer is available with PHP 4.3.0.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

`T_INCLUDE` (**integer**)

`T_INCLUDE_ONCE` (**integer**)

`T_EVAL` (**integer**)

`T_REQUIRE` (**integer**)

`T_REQUIRE_ONCE` (**integer**)

`T_LOGICAL_OR` (**integer**)

`T_LOGICAL_XOR` (**integer**)

`T_LOGICAL_AND` (**integer**)

`T_PRINT` (**integer**)

`T_PLUS_EQUAL` (**integer**)

`T_MINUS_EQUAL` (**integer**)

`T_MUL_EQUAL` (**integer**)

`T_DIV_EQUAL` (**integer**)

`T_CONCAT_EQUAL` (**integer**)

T_MOD_EQUAL (integer)

T_AND_EQUAL (integer)

T_OR_EQUAL (integer)

T_XOR_EQUAL (integer)

T_SL_EQUAL (integer)

T_SR_EQUAL (integer)

T_BOOLEAN_OR (integer)

T_BOOLEAN_AND (integer)

T_IS_EQUAL (integer)

T_IS_NOT_EQUAL (integer)

T_IS_IDENTICAL (integer)

T_IS_NOT_IDENTICAL (integer)

T_IS_SMALLER_OR_EQUAL (integer)

T_IS_GREATER_OR_EQUAL (integer)

T_SL (integer)

T_SR (integer)

T_INC (integer)

T_DEC (integer)

T_INT_CAST (integer)

T_DOUBLE_CAST (integer)

T_STRING_CAST (integer)

T_ARRAY_CAST (integer)

T_OBJECT_CAST (integer)

T_BOOL_CAST (integer)

T_UNSET_CAST (integer)

T_NEW (integer)

T_EXIT (integer)

T_IF (integer)

T_ELSEIF (integer)

T_ELSE (integer)

T_ENDIF (integer)

T_LNUMBER (integer)

T_DNUMBER (integer)

T_STRING (integer)

T_STRING_VARNAME (integer)

T_VARIABLE (integer)

T_NUM_STRING (integer)

T_INLINE_HTML (integer)

T_CHARACTER (integer)

T_BAD_CHARACTER (integer)

T_ENCAPSED_AND_WHITESPACE (integer)

T_CONSTANT_ENCAPSED_STRING (integer)

T_ECHO (integer)

T_DO (integer)

T_WHILE (integer)

T_ENDWHILE (integer)

T_FOR (integer)

T_ENDFOR (integer)

T_FOREACH (integer)

T_ENDFOREACH (integer)

T_DECLARE (integer)

T_ENDDECLARE (integer)

T_AS (integer)

T_SWITCH (integer)

T_ENDSWITCH (integer)

T_CASE (integer)

T_DEFAULT (integer)

T_BREAK (integer)

T_CONTINUE (integer)

T_OLD_FUNCTION (integer)

T_OLD_FUNCTION is not defined in PHP 5.

T_FUNCTION (integer)

T_CONST (integer)

T_RETURN (integer)

T_USE (integer)

T_GLOBAL (integer)

T_STATIC (integer)

T_VAR (integer)

T_UNSET (integer)

T_ISSET (integer)

T_EMPTY (integer)

T_CLASS (integer)

T_EXTENDS (integer)

T_OBJECT_OPERATOR (integer)

T_DOUBLE_ARROW (integer)

T_LIST (integer)

T_ARRAY (integer)

T_LINE (integer)

T_FILE (integer)

T_COMMENT (integer)

T_ML_COMMENT (integer)

T_ML_COMMENT is not defined in PHP 5. All comments in PHP 5 are of token **T_COMMENT**.

T_DOC_COMMENT (integer)

T_DOC_COMMENT was introduced in PHP 5.

T_OPEN_TAG (integer)

T_OPEN_TAG_WITH_ECHO (integer)

T_CLOSE_TAG (integer)

T_WHITESPACE (integer)

T_START_HEREDOC (integer)

T_END_HEREDOC (integer)

T_DOLLAR_OPEN_CURLY_BRACES (integer)

T_CURLY_OPEN (integer)

T_PAAMAYIM_NEKUDOTAYIM (integer)

T_DOUBLE_COLON (integer)

T_INTERFACE (integer)

PHP 5 only.

T_IMPLEMENTS (integer)

PHP 5 only.

T_CLASS_C (integer)

PHP 5 only.

T_FUNC_C (integer)

PHP 5 only.

T_METHOD_C (integer)

PHP 5 only.

T_ABSTRACT (integer)

PHP 5 only.

T_CATCH (integer)

PHP 5 only.

T_FINAL (integer)

PHP 5 only.

T_INSTANCEOF (integer)

PHP 5 only.

T_PRIVATE (integer)

PHP 5 only.

T_PROTECTED (integer)

PHP 5 only.

T_PUBLIC (integer)

PHP 5 only.

T_THROW (integer)

PHP 5 only.

T_TRY (integer)

PHP 5 only.

T_CLONE (integer)

PHP 5 only.

Ejemplos

Here is a simple example PHP scripts using the tokenizer that will read in a PHP file, strip all comments from the source and print the pure code only.

Ejemplo 1. Strip comments with the tokenizer

```php
<?php
/*
 * T_ML_COMMENT does not exist in PHP 5.
 * The following three lines define it in order to
 * preserve backwards compatibility.
 *
 * The next two lines define the PHP 5 only T_DOC_COMMENT,
 * which we will mask as T_ML_COMMENT for PHP 4.
 */
if (!defined('T_ML_COMMENT')) {
    define('T_ML_COMMENT', T_COMMENT);
} else {
    define('T_DOC_COMMENT', T_ML_COMMENT);
}

$source = file_get_contents('example.php');
$tokens = token_get_all($source);

foreach ($tokens as $token) {
    if (is_string($token)) {
        // simple 1-character token
        echo $token;
    } else {
        // token array
        list($id, $text) = $token;

        switch ($id) {
            case T_COMMENT:
            case T_ML_COMMENT: // we've defined this
            case T_DOC_COMMENT: // and this
                // no action on comments
                break;

            default:
                // anything else -> output "as is"
                echo $text;
                break;
        }
    }
}
?>
```

Tabla de contenidos

token_get_all

(PHP 4 >= 4.2.0, PHP 5)

token_get_all -- Split given source into PHP tokens

Descripción

array **token_get_all** (string source)

token_get_all() parses the given *source* string into PHP language tokens using the Zend engine's lexical scanner.

For a list of parser tokens, see Apéndice P, or use **token_name()** to translate a token value into its string representation.

Lista de parámetros

source

The PHP source to parse.

Valores retornados

An array of token identifiers. Each individual token identifier is either a single character (i.e.: *;*, ., >, *!*, etc...), or a two element array containing the token index in element 0, and the string content of the original token in element 1.

Ejemplos

Ejemplo 1. token_get_all() examples

```
<?php
$tokens = token_get_all('<?php'); // => array(array(T_OPEN_TAG, '<?'));
$tokens = token_get_all('<?php echo; ?>'); /* => array(
                                                  array(T_OPEN_TAG, '<?php'),
                                                  array(T_ECHO, 'echo'),
                                                  ';',
                                                  array(T_CLOSE_TAG, '?>') ); */

/* Note in the following example that the string is parsed as T_INLINE_HTML
   rather than the otherwise expected T_COMMENT (T_ML_COMMENT in PHP <5).
   This is because no open/close tags were used in the "code" provided.
   This would be ñalent to putting a comment outside of <?php ?> tags in a normal file.
$tokens = token_get_all('/* comment */'); // => array(array(T_INLINE_HTML, '/* comment
?>
```

token_name

(PHP 4 >= 4.2.0, PHP 5)

token_name -- Get the symbolic name of a given PHP token

Descripción

string **token_name** (int token)

token_name() gets the symbolic name for a PHP *token* value.

Lista de parámetros

token

The token value.

Valores retornados

The symbolic name of the given *token*. The returned name returned matches the name of the matching token constant.

Ejemplos

Ejemplo 1. token_name() example

```
<?php
// 260 is the token value for the T_REQUIRE token
echo token_name(260);        // -> "T_REQUIRE"

// a token constant maps to its own name
echo token_name(T_FUNCTION); // -> "T_FUNCTION"
?>
```

Ver también

List of Parser Tokens

CXXVII. ODBC functions

Tabla de contenidos

odbc_autocommit -- Interruptor de comportamiento de auto-entrega
odbc_binmode -- Manejo de campos de datos binarios
odbc_close_all -- Cierra todas las conexiones ODBC
odbc_close -- Cierra una conexion ODBC
odbc_columnprivileges -- Returns a result identifier that can be used to fetch a list of columns and associated privileges
odbc_columns -- Lists the column names in specified tables
odbc_commit -- Entrega una transaccion ODBC
odbc_connect -- Conecta a una fuente de datos
odbc_cursor -- Toma un nombre de cursor
odbc_data_source -- Returns information about a current connection
odbc_do -- sinonimo de **odbc_exec()**
odbc_error -- Get the last error code
odbc_errormsg -- Get the last error message
odbc_exec -- Prepara o ejecuta una declaracion SQL
odbc_execute -- ejecuta una declaracion preparada
odbc_fetch_array -- Fetch a result row as an associative array
odbc_fetch_into -- Busca un registro de resutados dentro de un vector
odbc_fetch_object -- Fetch a result row as an object
odbc_fetch_row -- Busca un registro
odbc_field_len -- Da la longitud de un campo
odbc_field_name -- Devuelve el nombre de campo
odbc_field_num -- Devuelve el numero de campo
odbc_field_precision -- Synonym for **odbc_field_len()**
odbc_field_scale -- Get the scale of a field
odbc_field_type -- Tipo de datos de un campo
odbc_foreignkeys -- Returns a list of foreign keys in the specified table or a list of foreign keys in other tables that refer to the primary key in the specified table
odbc_free_result -- recursos libres asociados con un resultado
odbc_gettypeinfo -- Returns a result identifier containing information about data types supported by the data source
odbc_longreadlen -- manejo de LONGITUD de columnas
odbc_next_result -- Checks if multiple results are available
odbc_num_fields -- numero de campos de un resultado
odbc_num_rows -- Numero de campos en un resultado
odbc_pconnect -- Abre una conexion permanente de base de datos

odbc_prepare -- Prepara una declaracion para su ejecucion
odbc_primarykeys -- Returns a result identifier that can be used to fetch the column names that comprise the primary key for a table
odbc_procedurecolumns -- Retrieve information about parameters to procedures
odbc_procedures -- Get the list of procedures stored in a specific data source
odbc_result_all -- Print result as HTML table
odbc_result -- coge informacion de un campo
odbc_rollback -- Volver a pasar una transacion
odbc_setoption -- Ajusta la configuracion de ODBC. Devuelve **FALSE** en caso de error, en otro caso **TRUE**.
odbc_specialcolumns -- Returns either the optimal set of columns that uniquely identifies a row in the table or columns that are automatically updated when any value in the row is updated by a transaction
odbc_statistics -- Retrieve statistics about a table
odbc_tableprivileges -- Lists tables and the privileges associated with each table
odbc_tables -- Get the list of table names stored in a specific data source

odbc_autocommit

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

odbc_autocommit -- Interruptor de comportamiento de auto-entrega

Descripcion

int **odbc_autocommit** (int connection_id [, int OnOff])

Sin el parametro *OnOff*, esta funcion devuelve el estado de auto-entrega para *connection_id*. Devuelve **TRUE** si auto-entrega esta habilitado, y **FALSE** si no lo esta o ha ocurrido un error.

Si *OnOff* es **TRUE**, auto-entrega esta activado, si es **FALSE** auto-entrega esta desactivado. Devuelve **TRUE** cuando se cumple, **FALSE** cuando falla.

Por defecto, auto-entrega es para una conexion. Desabilitar auto-entrega es como comenzar una transaccion.

Ver tambien **odbc_commit()** y **odbc_rollback()**.

odbc_binmode

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

odbc_binmode -- Manejo de campos de datos binarios

Descripcion

int **odbc_binmode** (int result_id, int mode)

(Elementos afectados ODBC SQL: BINARY, VARBINARY, LONGVARBINARY)

- ODBC_BINMODE_PASSTHRU: Paso a traves de datos binarios

- ODBC_BINMODE_RETURN: Devuelve como es
- ODBC_BINMODE_CONVERT: Devuelve convertido en caracter

Cuando los datos binarios en SQL son convertidos a datos caracter en C, cada byte (8 bits) de datos fuente es representada como dos caracteres en ASCII. Esos caracteres son la representacion en ASCII de los numeros en su forma Hexadecimal. Por ejemplo, un 00000001 binario es convertido a *"01"* y un 11111111 binario es convertido a *"FF"*.

Tabla 1. Manejo de LONGVARBINARY

modo binario	longreadlen	resultado
ODBC_BINMODE_PASSTHRU	0	passthru
ODBC_BINMODE_RETURN	0	passthru
ODBC_BINMODE_CONVERT	0	passthru
ODBC_BINMODE_PASSTHRU	0	passthru
ODBC_BINMODE_PASSTHRU	>0	passthru
ODBC_BINMODE_RETURN	>0	Devuleve como es
ODBC_BINMODE_CONVERT	>0	Devuelve como caracter

Si usamos **odbc_fetch_into()**, passthru significara que una cadena vacia es devuelta por esas campos.

Si *result_id* es *0*, las definiciones se aplican por defecto para nuevos resultados.

> **Nota:** Por defecto, longreadlen es *4096* y el modo binario por defecto es *ODBC_BINMODE_RETURN*. El manejo de campos binarias largas tambien esta afectado por **odbc_longreadlen()**

odbc_close_all

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

odbc_close_all -- Cierra todas las conexiones ODBC

Descripcion

void **odbc_close_all** (void)

odbc_close_all() cerrara todas las conexiones a servidor(es) de bases de datos.

> **Nota:** Esta funcion fallara si hay transacciones abiertas sobre esta conexion. La conexion quedara abierta en ese caso.

odbc_close

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

odbc_close -- Cierra una conexion ODBC

Descripcion

void **odbc_close** (int connection_id)

odbc_close() cerrara la conexion al servidor de bases datos asociado con el identificador de conexion dado.

> **Nota:** Esta funcion fallara si hay transacciones abiertas sobre esta conexion. La conexion quedara abierta en ese caso.

odbc_columnprivileges

(PHP 4 , PHP 5)

odbc_columnprivileges -- Returns a result identifier that can be used to fetch a list of columns and associated privileges

Description

resource **odbc_columnprivileges** (resource connection_id, string qualifier, string owner, string table_name, string column_name)

Lists columns and associated privileges for the given table. Returns an ODBC result identifier or **`FALSE`** on failure.

The result set has the following columns:

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- GRANTOR
- GRANTEE
- PRIVILEGE
- IS_GRANTABLE

The result set is ordered by TABLE_QUALIFIER, TABLE_OWNER and TABLE_NAME.

The *column_name* argument accepts search patterns ('%' to match zero or more characters and '_' to match a single character).

odbc_columns

(PHP 4 , PHP 5)

odbc_columns -- Lists the column names in specified tables

Description

resource **odbc_columns** (resource connection_id [, string qualifier [, string schema [, string table_name [, string column_name]]]])

Lists all columns in the requested range. Returns an ODBC result identifier containing the information or **FALSE** on failure.

The result set has the following columns:

- TABLE_QUALIFIER
- TABLE_SCHEM
- TABLE_NAME
- COLUMN_NAME
- DATA_TYPE
- TYPE_NAME
- PRECISION
- LENGTH
- SCALE
- RADIX
- NULLABLE
- REMARKS

The result set is ordered by TABLE_QUALIFIER, TABLE_SCHEM and TABLE_NAME.

The *schema*, *table_name* and *column_name* arguments accept search patterns ('%' to match zero or more characters and '_' to match a single character).

See also **odbc_columnprivileges()** to retrieve associated privileges.

odbc_commit

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

odbc_commit -- Entrega una transaccion ODBC

Descripcion

int **odbc_commit** (int connection_id)

Devuelve: **`TRUE`** si la operacion se realiza con exito, **`FALSE`** si falla. Todas las transacciones pendientes sobre *connection_id* son entregadas.

odbc_connect

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

odbc_connect -- Conecta a una fuente de datos

Descripcion

int **odbc_connect** (string dsn, string user, string password [, int cursor_type])

Devuelve una conexion ODBC id, o 0 (**`FALSE`**) cuando ocurre un error.

La conexion id devuelta por estas funciones es necesaria para otras funciones ODBC. Se pueden tener multiples conexiones abiertas a la vez. El opcional cuarto parametro asigna el tipo de cursor que va a ser usado para esta conexion. Este parametro normalmente no es necesario, pero puede ser util para trabajar sobre problemas con algunos drivers ODBC.

Con algunos drivers ODBC, si ejecutamos un procedimiento complejo, este puede fallar con un error similar a: "Cannot open a cursor on a stored procedure that has anything other than a single select statement in it". Usando SQL_CUR_USE_ODBC se puede evitar ese error. Algunos drivers tampoco soportan el parametro row_number en **odbc_fetch_row()**. SQL_CUR_USE_ODBC tambien podria ayudar en ese caso.

Las siguientes constantes son definidas por tipos de cursor:

- SQL_CUR_USE_IF_NEEDED
- SQL_CUR_USE_ODBC
- SQL_CUR_USE_DRIVER
- SQL_CUR_DEFAULT

Para conexiones persistentes ver **odbc_pconnect()**.

odbc_cursor

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

odbc_cursor -- Toma un nombre de cursor

Descripcion

string **odbc_cursor** (int result_id)

odbc_cursor devolvera un nombre de cursor para el result_id dado.

odbc_data_source

(PHP 4 >= 4.3.0, PHP 5)

odbc_data_source -- Returns information about a current connection

Description

array **odbc_data_source** (resource connection_id, int fetch_type)

Returns **FALSE** on error, and an array upon success.

This function will return the list of available DNS (after calling it several times). The *connection_id* is required to be a valid ODBC connection. The *fetch_type* can be one of two constant types: SQL_FETCH_FIRST, SQL_FETCH_NEXT. Use SQL_FETCH_FIRST the first time this function is called, thereafter use the SQL_FETCH_NEXT.

odbc_do

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

odbc_do -- sinonimo de **odbc_exec()**

Descripcion

string **odbc_do** (int conn_id, string query)

odbc_do ejecutara una consulta (query) sobre la conexion dada

odbc_error

(PHP 4 >= 4.0.5, PHP 5)

odbc_error -- Get the last error code

Description

string **odbc_error** ([resource connection_id])

Returns a six-digit ODBC state, or an empty string if there has been no errors. If *connection_id* is specified, the last state of that connection is returned, else the last state of any connection is returned.

This function returns meaningful value only if last odbc query failed (i.e. **odbc_exec()** returned **FALSE**).

See also: **odbc_errormsg()** and **odbc_exec()**.

odbc_errormsg

(PHP 4 >= 4.0.5, PHP 5)

odbc_errormsg -- Get the last error message

Description

string **odbc_errormsg** ([resource connection_id])

Returns a string containing the last ODBC error message, or an empty string if there has been no errors. If *connection_id* is specified, the last state of that connection is returned, else the last state of any connection is returned.

This function returns meaningful value only if last odbc query failed (i.e. **odbc_exec()** returned **FALSE**).

See also: **odbc_error()** and **odbc_exec()**.

odbc_exec

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

odbc_exec -- Prepara o ejecuta una declaracion SQL

Descripcion

int **odbc_exec** (int connection_id, string query_string)

Devuelve **FALSE** en caso de error. Devuelve un indetificador ODBC si el comando SQL fue ejecutado satisfactoriamente.

odbc_exec() enviara una declaracion SQL al servidor de bases de datos specificado por *connection_id*. Este parametro debe ser un indetificador valido devuelto por **odbc_connect()** o **odbc_pconnect()**.

Ver tambien: **odbc_prepare()** y **odbc_execute()** para ejecucion multiple de declaraciones SQL.

odbc_execute

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

odbc_execute -- ejecuta una declaracion preparada

Descripcion

int **odbc_execute** (int result_id [, array parameters_array])

Ejecuta uan declaracion preparada con **odbc_prepare()**. Devuelve **TRUE** cuando la ejecucion es satisfactoria, **FALSE** en otro caso. Introducir el vector *arameters_array* solo es necesario si realmente tenemos parametros en la declaracion.

odbc_fetch_array

(PHP 4 >= 4.0.2, PHP 5)

odbc_fetch_array -- Fetch a result row as an associative array

Description

array **odbc_fetch_array** (resource result [, int rownumber])

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

odbc_fetch_into

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

odbc_fetch_into -- Busca un registro de resutados dentro de un vector

Descripcion

int **odbc_fetch_into** (int result_id [, int rownumber, array result_array])

Devuelve el numero de campos en el resultado; **FALSE** on error. *result_array* debe ser pasado por referencia, pero puede ser de cualquier tipo, desde este sera convertido a tipo vector. El vector contendra el valor de campo inicial empezando en indice de vector 0.

odbc_fetch_object

(PHP 4 >= 4.0.2, PHP 5)

odbc_fetch_object -- Fetch a result row as an object

Description

object **odbc_fetch_object** (resource result [, int rownumber])

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

odbc_fetch_row

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

odbc_fetch_row -- Busca un registro

Descripcion

int **odbc_fetch_row** (int result_id [, int row_number])

Si **odbc_fetch_row()** fue succesful (there was a row), **TRUE** is returned. If there are no more rows, **FALSE** is returned.

odbc_fetch_row() busca un registro de datos que fue devuelta por **odbc_do()** / **odbc_exec()**. Despues de que **odbc_fetch_row()** sea llamado, se puede acceder a los campos de este registro con **odbc_result()**.

Si no se especifica *row_number*, **odbc_fetch_row()** intentara buscar el siguiente registro en los resultados. Lamar a **odbc_fetch_row()** con o sin *row_number* puede ser mezclado.

Para pasar a traves del resultado mas de una vez, se puede llamar a **odbc_fetch_row()** con *row_number* 1, y despues continuar haciendo **odbc_fetch_row()** sin *row_number* para revisar el resultado. Si un driver no admitiese busquedas de registros por numero, el parametro *row_number* seria ignorado.

odbc_field_len

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

odbc_field_len -- Da la longitud de un campo

Descripcion

int **odbc_field_len** (int result_id, int field_number)

odbc_field_len() devolvera la longitud de un campo referenciado por numero en un identificador ODBC La numeracion de campos comienza en 1.

odbc_field_name

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

odbc_field_name -- Devuelve el nombre de campo

Descripcion

string **odbc_field_name** (int result_id, int field_number)

odbc_field_name() devolvera el nombre del campo almacenado en el numero de campo elegido dentro del identificador ODBC. La numeracion de campos comienza en 1. En caso de error devolveria **FALSE**.

odbc_field_num

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

odbc_field_num -- Devuelve el numero de campo

Descripcion

int **odbc_field_num** (int result_id, string field_name)

odbc_field_num() devolvera el numero de campo que corresponda con el campo llamado en el identificador ODBC. La numeracion de campos comienza en 1. En caso de error devolveria **FALSE**.

odbc_field_precision

(PHP 4 , PHP 5)

odbc_field_precision -- Synonym for **odbc_field_len()**

Description

int **odbc_field_precision** (resource result_id, int field_number)

odbc_field_precision() will return the precision of the field referenced by number in the given ODBC result identifier.

See also: **odbc_field_scale()** to get the scale of a floating point number.

odbc_field_scale

(PHP 4 , PHP 5)

odbc_field_scale -- Get the scale of a field

Description

int **odbc_field_scale** (resource result_id, int field_number)

odbc_field_scale() will return the scale of the field referenced by number in the given ODBC result identifier.

odbc_field_type

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

odbc_field_type -- Tipo de datos de un campo

Descripcion

string **odbc_field_type** (int result_id, int field_number)

odbc_field_type() Devolvera el tipo SQL de un campo referenciado por numero en el identificador ODBC. identifier. La numeracion de campos comienza en 1.

odbc_foreignkeys

(PHP 4 , PHP 5)

odbc_foreignkeys -- Returns a list of foreign keys in the specified table or a list of foreign keys in other tables that refer to the primary key in the specified table

Description

resource **odbc_foreignkeys** (resource connection_id, string pk_qualifier, string pk_owner, string pk_table, string fk_qualifier, string fk_owner, string fk_table)

odbc_foreignkeys() retrieves information about foreign keys. Returns an ODBC result identifier or **`FALSE`** on failure.

The result set has the following columns:

- PKTABLE_QUALIFIER
- PKTABLE_OWNER
- PKTABLE_NAME
- PKCOLUMN_NAME
- FKTABLE_QUALIFIER
- FKTABLE_OWNER
- FKTABLE_NAME
- FKCOLUMN_NAME
- KEY_SEQ
- UPDATE_RULE
- DELETE_RULE

- FK_NAME
- PK_NAME

If *pk_table* contains a table name, **odbc_foreignkeys()** returns a result set containing the primary key of the specified table and all of the foreign keys that refer to it.

If *fk_table* contains a table name, **odbc_foreignkeys()** returns a result set containing all of the foreign keys in the specified table and the primary keys (in other tables) to which they refer.

If both *pk_table* and *fk_table* contain table names, **odbc_foreignkeys()** returns the foreign keys in the table specified in *fk_table* that refer to the primary key of the table specified in *pk_table*. This should be one key at most.

odbc_free_result

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

odbc_free_result -- recursos libres asociados con un resultado

Descripcion

int **odbc_free_result** (int result_id)

Always returns **TRUE**.

odbc_free_result() solo necesita ser llamado en caso de preocupacion por demasiado uso de memoria cuando se ejecuta un script. Toda la memoria resultante quedara automaticamente liberada cuando el script finalice. Pero si es seguro que no se vaya a necesitar la informacion nada mas que en un script, se debera llamar a la funcion **odbc_free_result()**, y la memoria asociada con *result_id* sera liberada.

> **Nota:** Si la auto-entrega no esta activada la (ver **odbc_autocommit()**) y se ejecuta **odbc_free_result()** antes de la entrega, todo queda pendiente de las transaciones que esten en lista.

odbc_gettypeinfo

(PHP 4 , PHP 5)

odbc_gettypeinfo -- Returns a result identifier containing information about data types supported by the data source

Description

resource **odbc_gettypeinfo** (resource connection_id [, int data_type])

Retrieves information about data types supported by the data source. Returns an ODBC result identifier or **FALSE** on failure. The optional argument *data_type* can be used to restrict the information to a single data type.

The result set has the following columns:

- TYPE_NAME
- DATA_TYPE
- PRECISION
- LITERAL_PREFIX
- LITERAL_SUFFIX
- CREATE_PARAMS
- NULLABLE
- CASE_SENSITIVE
- SEARCHABLE
- UNSIGNED_ATTRIBUTE
- MONEY
- AUTO_INCREMENT
- LOCAL_TYPE_NAME
- MINIMUM_SCALE
- MAXIMUM_SCALE

The result set is ordered by DATA_TYPE and TYPE_NAME.

odbc_longreadlen

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

odbc_longreadlen -- manejo de LONGITUD de columnas

Descripcion

int **odbc_longreadlen** (int result_id, int length)

(ODBC SQL tipos relaccionados: LONG, LONGVARBINARY) El numero de bytes devueltos para PHP es controlado por el parametro length. Si es asignado a 0, la longitud del campo es pasado al cliente.

> **Nota:** El manejo de campos LONGVARBINARY tambien esta afectado por **odbc_binmode()**

odbc_next_result

(PHP 4 >= 4.0.5, PHP 5)

odbc_next_result -- Checks if multiple results are available

Description

bool **odbc_next_result** (resource result_id)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

odbc_num_fields

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

odbc_num_fields -- numero de campos de un resultado

Descripcion

int **odbc_num_fields** (int result_id)

odbc_num_fields() devolvera el numero de campos dentro de un ODBC. Esta funcion devolvera -1 en caso de error. El argumento es un identificador valido devuelto por **odbc_exec()**.

odbc_num_rows

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

odbc_num_rows -- Numero de campos en un resultado

Descripcion

int **odbc_num_rows** (int result_id)

odbc_num_rows() devolvera el numero de registros de un ODBC. Esta funcion devolvera -1 en caso de error. Para declaraciones INSERT, UPDATE y DELETE **odbc_num_rows()** devolvera el numero de registros afectados. Para una clausula SELECT esta *puede* ser el numero de registros permitidos.

Nota: El uso de **odbc_num_rows()** para determinar el numero de registros permitidos despues de un SELECT devolvera -1.

odbc_pconnect

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

odbc_pconnect -- Abre una conexion permanente de base de datos

Descripcion

int **odbc_pconnect** (string dsn, string user, string password [, int cursor_type])

Devuelve un identificador de conexion ODBC o 0 (**FALSE**) en caso de error. Esta funcion es **odbc_connect()**, excepto que la conexion no sea realmente cerrada cuando el script ha finalizado. Las respuestas futuras para una conexion con la misma combinacion *dsn*, *user*, *password* (via **odbc_connect()** y **odbc_pconnect()**) puede reusar la conexion permanente.

> **Nota:** Las conexiones permanentes no tienen efecto si PHP es usado como programa CGI.

Para informacion acerca del paramentor opcional cursor_type ver la funcion **odbc_connect()**. Para mas informacion sobre conexiones permanentes, ir al apartado PHP FAQ.

odbc_prepare

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

odbc_prepare -- Prepara una declaracion para su ejecucion

Descripcion

int **odbc_prepare** (int connection_id, string query_string)

Devuelve **FALSE** en caso de error.

Devuelve un identificador ODBC si el comando SQL esta preparado. El identificador resultante puede ser usado mas tarde para ejecutar la declaracion con **odbc_execute()**.

odbc_primarykeys

(PHP 4 , PHP 5)

odbc_primarykeys -- Returns a result identifier that can be used to fetch the column names that comprise the primary key for a table

Description

resource **odbc_primarykeys** (resource connection_id, string qualifier, string owner, string table)

Returns the column names that comprise the primary key for a table. Returns an ODBC result identifier or **FALSE** on failure.

The result set has the following columns:

- TABLE_QUALIFIER

- TABLE_OWNER
- TABLE_NAME
- COLUMN_NAME
- KEY_SEQ
- PK_NAME

odbc_procedurecolumns

(PHP 4 , PHP 5)

odbc_procedurecolumns -- Retrieve information about parameters to procedures

Description

resource **odbc_procedurecolumns** (resource connection_id [, string qualifier, string owner, string proc, string column])

Returns the list of input and output parameters, as well as the columns that make up the result set for the specified procedures. Returns an ODBC result identifier or **`FALSE`** on failure.

The result set has the following columns:

- PROCEDURE_QUALIFIER
- PROCEDURE_OWNER
- PROCEDURE_NAME
- COLUMN_NAME
- COLUMN_TYPE
- DATA_TYPE
- TYPE_NAME
- PRECISION
- LENGTH
- SCALE
- RADIX
- NULLABLE
- REMARKS

The result set is ordered by PROCEDURE_QUALIFIER, PROCEDURE_OWNER, PROCEDURE_NAME and COLUMN_TYPE.

The *owner*, *proc* and *column* arguments accept search patterns ('%' to match zero or more characters and '_' to match a single character).

odbc_procedures

(PHP 4 , PHP 5)

odbc_procedures -- Get the list of procedures stored in a specific data source

Description

resource **odbc_procedures** (resource connection_id [, string qualifier, string owner, string name])

Lists all procedures in the requested range. Returns an ODBC result identifier containing the information or **FALSE** on failure.

The result set has the following columns:

- PROCEDURE_QUALIFIER
- PROCEDURE_OWNER
- PROCEDURE_NAME
- NUM_INPUT_PARAMS
- NUM_OUTPUT_PARAMS
- NUM_RESULT_SETS
- REMARKS
- PROCEDURE_TYPE

The *owner* and *name* arguments accept search patterns ('%' to match zero or more characters and '_' to match a single character).

odbc_result_all

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

odbc_result_all -- Print result as HTML table

Descripcion

int **odbc_result_all** (int result_id [, string format])

En caso de error, como resultado, devuelve **FALSE**.

odbc_result_all() Imprimira todos los registros de un identificador prducidoo por **odbc_exec()**. El resultado es impreso en una tabla formato HTML. Con el argumento de cadena opcional *format*, ademas, todas los formatos de tablas pueden ser realizadas.

odbc_result

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

odbc_result -- coge informacion de un campo

Descripcion

string **odbc_result** (int result_id, mixed field)

Devuelve el contenido de un campo.

field puede ser cualquier contenido del campo que queramos; o puede ser una cadena que contenga el nombre del campo; Por ejemplo:

```
$item_3 = odbc_result($Query_ID, 3 );
     $item_val = odbc_result($Query_ID, "val");
```

La primera sentencia **odbc_result()** devuelve el valor del tercer campo detro del registro actual de la cola resultante. La segunda funcion llama a **odbc_result()** y devuelve el valor de un campo cuyo nombre es "val" en el registro actual de la cola resultante. Ocurre un error si un numero de columna para un campo es menor que uno o excede el numero de campos en el registro actual. Similarmente, ocurre un error si un campo con un nombre que no sea uno de los nombres de campo de una talba o tablas que sea o sean encoladas.

Los indices de campo comienzan en 1. Recordando el metodo binario de campos con gran informacion, es devuleto con referencia a **odbc_binmode ()** y **odbc_longreadlen()**.

odbc_rollback

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

odbc_rollback -- Volver a pasar una transacion

Descripcion

int **odbc_rollback** (int connection_id)

Vuelve a pasar todas las declaraciones pendientes *connection_id*. Devuelve **TRUE** cuando el resultado es satisfactorio, **FALSE** cuando no lo es.

odbc_setoption

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

odbc_setoption -- Ajusta la configuracion de ODBC. Devuelve **FALSE** en caso de error, en otro caso **TRUE**.

Descripcion

int **odbc_setoption** (int id, int function, int option, int param)

Esta funcion permite buscar las opciones ODBC para una conexion particular o consulta resultante. Esto esta escrito para trabajar sobre problemas en peculiaries drivers ODBC. Esta funcion Solo se deberia usar siendo un programador de ODBC y entendiendo los efectos que las opciones tendran. Debemos tener la certeza de que necesitamos una buena referencia de reference to explicar todas las diferentes opciones y valores que pueden ser usados. Las diferentes versiones de drivers soportan diferentes opciones.

Ya que los efectos pueden variar dependiendo del driver ODBC, deberiamos usar la function en scripts para ser hecho publico lo que permitira que sea fuertemente desalentado. Algunas opciones ODBC no estan permitidas para esta funcion porque debe ser configurada antes de que la conexion sea establecida o la consulta este preparada. Sin embargo, si un determinado trabajo hace la tarea de PHP, el jefe no contaria con nosotros para usar un producto comercial, esto es lo que realmente suele pasar.

Id es una coexion id o resultado id sobre la que cambiaremos la configuracion. Para SQLSetConnectOption(), esta es una conexion id. Para SQLSetStmtOption(), este es un resultado id.

function es la funcion ODBC a usar. El valor deberia ser 1 para SQLSetConnectOption() y 2 para SQLSetStmtOption().

Parmeter *option* es la opcion a configurar.

El parametro *param* es el valor para la escogida opcion *option*.

Ejemplo 1. Ejemplos ODBC Setoption

```
// 1. Option 102 of SQLSetConnectOption() is SQL_AUTOCOMMIT.
//    Value 1 of SQL_AUTOCOMMIT is SQL_AUTOCOMMIT_ON.
//    Este ejemplo tiene el mismo efecto que
//    odbc_autocommit($conn, true);

odbc_setoption ($conn, 1, 102, 1);

// 2. Option 0 of SQLSetStmtOption() is SQL_QUERY_TIMEOUT.
//    Este ejemplo asigna el tiempo de espera de la consulta a 30 segundos.

$result = odbc_prepare ($conn, $sql);
odbc_setoption ($result, 2, 0, 30);
odbc_execute ($result);
```

odbc_specialcolumns

(PHP 4 , PHP 5)

odbc_specialcolumns -- Returns either the optimal set of columns that uniquely identifies a row in the table or columns that are automatically updated when any value in the row is updated by a transaction

Description

resource **odbc_specialcolumns** (resource connection_id, int type, string qualifier, string owner, string table, int scope, int nullable)

When the type argument is SQL_BEST_ROWID, **odbc_specialcolumns()** returns the column or columns that uniquely identify each row in the table.

When the type argument is SQL_ROWVER, **odbc_specialcolumns()** returns the column or columns in the specified table, if any, that are automatically updated by the data source when any value in the row is updated by any transaction.

Returns an ODBC result identifier or **FALSE** on failure.

The result set has the following columns:

- SCOPE
- COLUMN_NAME
- DATA_TYPE
- TYPE_NAME
- PRECISION
- LENGTH
- SCALE
- PSEUDO_COLUMN

The result set is ordered by SCOPE.

odbc_statistics

(PHP 4 , PHP 5)

odbc_statistics -- Retrieve statistics about a table

Description

resource **odbc_statistics** (resource connection_id, string qualifier, string owner, string table_name, int unique, int accuracy)

Get statistics about a table and its indexes. Returns an ODBC result identifier or **FALSE** on failure.

The result set has the following columns:

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- NON_UNIQUE

- INDEX_QUALIFIER
- INDEX_NAME
- TYPE
- SEQ_IN_INDEX
- COLUMN_NAME
- COLLATION
- CARDINALITY
- PAGES
- FILTER_CONDITION

The result set is ordered by NON_UNIQUE, TYPE, INDEX_QUALIFIER, INDEX_NAME and SEQ_IN_INDEX.

odbc_tableprivileges

(PHP 4 , PHP 5)

odbc_tableprivileges -- Lists tables and the privileges associated with each table

Description

resource **odbc_tableprivileges** (resource connection_id, string qualifier, string owner, string name)

Lists tables in the requested range and the privileges associated with each table. Returns an ODBC result identifier or **`FALSE`** on failure.

The result set has the following columns:

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- GRANTOR
- GRANTEE
- PRIVILEGE
- IS_GRANTABLE

The result set is ordered by TABLE_QUALIFIER, TABLE_OWNER and TABLE_NAME.

The *owner* and *name* arguments accept search patterns ('%' to match zero or more characters and '_' to match a single character).

odbc_tables

(PHP 3>= 3.0.17, PHP 4 , PHP 5)

odbc_tables -- Get the list of table names stored in a specific data source

Description

resource **odbc_tables** (resource connection_id [, string qualifier [, string owner [, string name [, string types]]]])

Lists all tables in the requested range. Returns an ODBC result identifier containing the information or **`FALSE`** on failure.

The result set has the following columns:

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- TABLE_TYPE
- REMARKS

The result set is ordered by TABLE_TYPE, TABLE_QUALIFIER, TABLE_OWNER and TABLE_NAME.

The *owner* and *name* arguments accept search patterns ('%' to match zero or more characters and '_' to match a single character).

To support enumeration of qualifiers, owners, and table types, the following special semantics for the *qualifier*, *owner*, *name*, and *table_type* are available:

- If *qualifier* is a single percent character (%) and *owner* and *name* are empty strings, then the result set contains a list of valid qualifiers for the data source. (All columns except the TABLE_QUALIFIER column contain NULLs.)
- If *owner* is a single percent character (%) and *qualifier* and *name* are empty strings, then the result set contains a list of valid owners for the data source. (All columns except the TABLE_OWNER column contain NULLs.)
- If *table_type* is a single percent character (%) and *qualifier*, *owner* and *name* are empty strings, then the result set contains a list of valid table types for the data source. (All columns except the TABLE_TYPE column contain NULLs.)

If *table_type* is not an empty string, it must contain a list of comma-separated values for the types of interest; each value may be enclosed in single quotes (') or unquoted. For example, "'TABLE','VIEW'" or "TABLE, VIEW". If the data source does not support a specified table type,

odbc_tables() does not return any results for that type.

See also **odbc_tableprivileges()** to retrieve associated privileges.

CXXVIII. Funciones de URL

Introducción

Tratamiento de cadenas de URL: codificación, decodificación y procesamiento.

Requirimientos

No se necesitan bibliotecas externas para construir esta extensión

---

Instalación

No se necesita ninguna instalación para usar estas funciones, son parte del núcleo de PHP.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Esta extensión no tiene ninguna constante definida.

Tabla de contenidos

urlencode -- Codifica una URL en una cadena de texto

base64_decode

(PHP 3, PHP 4 , PHP 5)

base64_decode -- decodifica datos cifrados con MIME base64

Descripción

string **base64_decode** (string datos_cifrados)

base64_decode() decodifica *datos_cifrados* y devuelve los datos originales. Los datos devueltos pueden ser binarios.

Vea también: **base64_encode()**, RFC-2045 sección 6.8.

base64_encode

(PHP 3, PHP 4 , PHP 5)

base64_encode -- Codifica datos en MIME base64

Descripción

string **base64_encode** (string datos)

base64_encode() devuelve *datos* cifrados en base64. Esta codificación está pensada para que los datos binarios sobrevivan al transporte a través de capas que no son de 8 bits, como por ejemplo los cuerpos de los mensajes de correo.

Los datos codificados con Base64 ocupan aproximadamente un 33% más de espacio que los datos originales.

Vea también: **base64_decode()**, **chunk_split()**, RFC-2045 sección 6.8.

get_headers

(PHP 5)

get_headers -- Recupera todas las cabeceras enviadas por el servidor en respuesta a una petición HTTP

Descripción

array **get_headers** (string url [, bool formato])

get_headers() devuelve una matriz con las cabeceras enviadas por el servidor en respuesta a una petición HTTP.

Si el parámetro opcional *formato* es **TRUE**, **get_headers()** interpreta la respuesta y define las claves de la matriz.

Ejemplo 1. Ejemplo de get_headers()

```
<?php
$url = 'http://www.example.com';

print_r(get_headers($url));

print_r(get_headers($url, true));
?>
```

El ejemplo anterior producirá una salida como:

```
Array
(
    [0] => HTTP/1.1 200 OK
    [1] => Date: Sat, 29 May 2004 12:28:13 GMT
    [2] => Server: Apache/1.3.27 (Unix)  (Red-Hat/Linux)
    [3] => Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
    [4] => ETag: "3f80f-1b6-3e1cb03b"
    [5] => Accept-Ranges: bytes
    [6] => Content-Length: 438
    [7] => Connection: close
    [8] => Content-Type: text/html
)

Array
(
    [0] => HTTP/1.1 200 OK
    [Date] => Sat, 29 May 2004 12:28:14 GMT
    [Server] => Apache/1.3.27 (Unix)  (Red-Hat/Linux)
    [Last-Modified] => Wed, 08 Jan 2003 23:11:55 GMT
    [ETag] => "3f80f-1b6-3e1cb03b"
    [Accept-Ranges] => bytes
    [Content-Length] => 438
    [Connection] => close
    [Content-Type] => text/html
)
```

get_meta_tags

(PHP 3>= 3.0.4, PHP 4 , PHP 5)

get_meta_tags -- Extrae todo el contenido de atributos de etiquetas meta de un archivo y devuelve una matriz

Descripción

array **get_meta_tags** (string nombre_archivo [, bool usar_ruta_inclusion])

Abre *nombre_archivo* y lo procesa línea por línea en busca de etiquetas <meta> en el archivo. Éste puede ser un archivo local o una URL. El procesamiento se detiene al encontrar *</head>*.

Definir *usar_ruta_inclusion* como **TRUE** producirá que PHP intente abrir el archivo a lo largo de la ruta de inclusión estándar, tal y como se define en la directiva include_path. Éste parámetro es usado para archivos locales, no URLs.

Ejemplo 1. Lo que procesa get_meta_tags()

```
<meta name="author" content="nombre">
<meta name="keywords" content="php documentacion">
<meta name="DESCRIPTION" content="un manual de php">
<meta name="geo.position" content="49.33;-86.59">
</head> <!-- el procesamiento se detiene aqui -->
```

(preste atención a los finales de línea, - PHP usa una función nativa para procesar la entrada, así que un archivo Mac no funcionará en Unix).

El valor de la propiedad name se convierte en la clave, el valor de contenido de la propiedad se convierte en el valor de la matriz devuelta, de modo que puede usar fácilmente funciones estándar de matrices para recorrerlo o acceder a valores sencillos. Los caracteres especiales en el valor de la propiedad name son sustituidos con '_', el resto es convertido a minúsculas. Si dos etiquetas meta tienen el mismo nombre, sólo se devuelve la última.

Ejemplo 2. Lo que devuelve get_meta_tags()

```
<?php
// Asumiendo que las anteriores etiquetas se encuentran en www.example.com
$etiquetas = get_meta_tags('http://www.example.com/');

// Note como las claves estan ahora en minusculas, y como . fue
// reemplazado con _ en la clave
echo $etiquetas['author'];       // nombre
echo $etiquetas['keywords'];     // php documentacion
echo $etiquetas['description'];  // un manual de php
echo $etiquetas['geo_position']; // 49.33;-86.59
?>
```

Nota: A partir de PHP 4.0.5, **get_meta_tags()** soporta los atributos HTML sin comillas.

Vea también **htmlentities()** y **urlencode()**.

http_build_query

(PHP 5)

http_build_query -- Generar una cadena de consulta codificada estilo URL

Descripción

string **http_build_query** (array datos_formulario [, string prefijo_numerico])

Genera una cadena de consulta codificada estilo URL a partir de la matriz asociativa (o indexada) dada. *datos_formulario* puede ser una matriz u objeto que contenga propiedades. Una matriz *datos_formulario* puede ser una estructura uni-dimensional sencilla, o una matriz de matrices (que a su vez puede contener otras matrices). Si se usan índices numéricos en la matriz base, y se provee un *prefijo_numerico*, éste será añadido al comienzo de los índices numéricos para aquellos elementos encontrados sólo en la matriz base. Esto es para permitir que se opere con nombres de variables legales cuando los datos sean decodificados por PHP u otra aplicación CGI más adelante.

Ejemplo 1. Uso simple de http_build_query()

```
<?php
$datos = array('foo'=>'bar',
               'baz'=>'boom',
               'vaca'=>'leche',
               'php'=>'procesador de hipertexto');

echo http_build_query($datos); // foo=bar&baz=boom&vaca=leche&php=procesador+de+hiperte

?>
```

Ejemplo 2. http_build_query() con elementos indexados numéricamente.

```
<?php
$datos = array('foo', 'bar', 'baz', 'boom', 'vaca' => 'leche',
               'php' => 'procesador de hipertexto');

echo http_build_query($datos);
/* Genera la salida:
      0=foo&1=bar&2=baz&3=boom&vaca=leche&php=procesador+de+hipertexto
 */

echo http_build_query($datos, 'mivar_');
/* Genera la salida:
      mivar_0=foo&mivar_1=bar&mivar_2=baz&mivar_3=boom&vaca=leche&php=procesador+de+hip
 */
?>
```

Ejemplo 3. http_build_query() con matrices complejas

```
<?php
$datos = array('usuario'=>array('nombre'=>'Bob Smith',
                                'edad'=>47,
                                'sexo'=>'M',
                                'fdm'=>'5/12/1956'),
               'pasatiempos'=>array('golf', 'opera', 'poker', 'rap'),
               'hijos'=>array('bobby'=>array('edad'=>12,
                                            'sexo'=>'M'),
                              'sally'=>array('edad'=>8,
                                            'sexo'=>'F')),
               'CEO');

echo http_build_query($datos, 'banderas_');
?>
```

esto generará la salida: (acotada por razones de legibilidad)

```
usuario[nombre]=Bob+Smith&usuario[edad]=47&usuario[sexo]=M&
usuario[fdm]=5%1F12%1F1956&pasatiempos[0]=golf&pasatiempos[1]=opera&
pasatiempos[2]=poker&pasatiempos[3]=rap&hijos[bobby][edad]=12&
hijos[bobby][sexo]=M&hijos[sally][edad]=8&hijos[sally][sexo]=F&banderas_0=CEO
```

> **Nota:** Sólo el elemento "CEO" indexado numéricamente en la matriz base recibió un prefijo. Los otros índices numéricos, encontrados bajo los pasatiempos, no requieren un prefijo tipo cadena para ser nombres legales de variables.

Ejemplo 4. Uso de http_build_query() con un objeto

```
<?php
class miClase {
  var $foo;
  var $baz;

  function miClase()
  {
    $this->foo = 'bar';
    $this->baz = 'boom';
  }
}

$datos = new miClase();

echo http_build_query($datos); // foo=bar&baz=boom

?>
```

Vea también: **parse_str()**, **parse_url()**, **urlencode()**, y **array_walk()**

parse_url

(PHP 3, PHP 4 , PHP 5)

parse_url -- Analiza una URL y devuelve sus componentes

Descripción

array **parse_url** (string url)

Esta función devuelve una matriz asociativa con los componentes de la URL que estén presentes. Si alguno de ellos no está presente no se creará su correspondiente clave.

- `scheme` - por ejemplo: http
- `host`
- `port`
- `user`
- `pass`
- `path`
- `query` - lo que hay después de *?*
- `fragment` - lo que hay después de #

Esta función **no** está destinada a validar la URL dada, solo devuelve las partes correspondientes de una URL. También se aceptan URLs parciales, **parse_url()** intenta interpretarlas correctamente.

> **Nota:** Esta función no funciona con URLs relativas al documento

Ejemplo 1. parse_url() example

```
$ php -r 'print_r(parse_url("http://usuario:contrasena@servidor/ruta?argumento=valor#an
Array
(
    [scheme] => http
    [host] => servidor
    [user] => usuario
    [pass] => contrasena
    [path] => /ruta
    [query] => argumento=valor
    [fragment] => ancla
)

$ php -r 'print_r(parse_url("http://servidor_invalido..dominio/"));'
Array
(
    [scheme] => http
    [host] => servidor_invalido..dominio
    [path] => /
)
```

Ver también **pathinfo()**, **parse_str()**, **dirname()**, y **basename()**.

rawurldecode

(PHP 3, PHP 4 , PHP 5)

rawurldecode -- Decodificar cadenas codificadas estilo URL

Descripción

string **rawurldecode** (string cadena)

Devuelve una cadena en done las con secuencias con signos de porcentaje (%) seguidos de dos dígitos hexadecimales, son reemplazados con caracteres literales.

Ejemplo 1. Ejemplo de rawurldecode()

```
<?php

echo rawurldecode('foo%20bar%40baz'); // foo bar@baz

?>
```

> **Nota: rawurldecode()** no decodifica los símbolos más ('+') como espacios. **urldecode()** lo hace.

Vea también **rawurlencode()**, **urldecode()** y **urlencode()**.

rawurlencode

(PHP 3, PHP 4 , PHP 5)

rawurlencode -- Codificar estilo URL de acuerdo al RFC 1738

Descripción

string **rawurlencode** (string cadena)

Devuelve una cadena en donde todos los caracteres no-alfanuméricos, excepto -_., son reemplazados con un signo de porcentaje (%) seguido de dos dígitos hexadecimales. Este es el tipo de codificación descrito en el RFC 1738 para evitar que caracteres literales sean interpretados como delimitadores de URL especiales, y para evitar que las URLs sean modificadas por medios de transmisión con conversiones de caracteres (como algunos sistemas de correo electrónico). Por ejemplo, si desea incluir una contraseña en una URL de FTP:

Ejemplo 1. Ejemplo 1 de rawurlencode()

```
<?php
echo '<a href="ftp://usuario:', rawurlencode('foo @+%/'),
     '@ftp.example.com/x.txt">';
?>
```

O, si pasa información en un componente PATH_INFO de la URL:

Ejemplo 2. Ejemplo 2 de rawurlencode()

```
<?php
echo '<a href="http://example.com/script_de_lista_de_departamentos/',
    rawurlencode('ventas y mercadeo/Miami'), '">';
?>
```

Vea también **rawurldecode()**, **urldecode()**, **urlencode()** y el documento RFC 1738.

urldecode

(PHP 3, PHP 4 , PHP 5)

urldecode -- decodifica URL-cifradas en una cadena de texto

Descripción

string **urldecode** (string cadena)

Decodifica cualquier *%##* cifrado en la cadena dada. Se devuelve la cadena decodificada.

Ejemplo 1. Ejemplo urldecode()

```
$a = split ('&', $querystring);
$i = 0;
while ($i < count ($a)) {
  $b = split ('=', $a [$i]);
  echo 'El valor para el parámetro ', htmlspecialchars (urldecode ($b [0])),
       ' es ', htmlspecialchars (urldecode ($b [1])), "<BR>";
  $i++;
}
```

Vea también **urlencode()**

urlencode

(PHP 3, PHP 4 , PHP 5)

urlencode -- Codifica una URL en una cadena de texto

Descripción

string **urlencode** (string cadena)

Devuelve una cadena en la que todos los carácteres no alfanuméricos excepto -_. han sido reemplazados con un signo de porcentaje (*%*) seguido por dos dígitos hexadecimales y los espacios han sido codificados como signos positivos (+). Está codificado de la misma manera que los datos que se envian desde un formulario WWW, es decir de la misma forma que el tipo *application/x-www-form-urlencoded*. Esto difiere del cifrado RFC1738 (vea **rawurlencode()**) en el que por razones históricas, los espacios son codificados como signos positivos (+). Esta función es conveniente para codificar una cadena de texto que va a ser usada como parte de una consulta de una URL, y es una forma adecuada de pasar variables a la página siguiente:

Ejemplo 1. Ejemplo urlencode()

```
echo '<A HREF="mycgi?foo=', urlencode ($userinput), '">';
```

Vea también **urldecode()**

CXXIX. Funciones de Variables

Introducción

Para más información sobre el modo en que se comportan las variables, vea la entrada Variables en la sección Referencia del Lenguaje del manual.

Requirimientos

No se necesitan bibliotecas externas para construir esta extensión

---

Instalación

No se necesita ninguna instalación para usar estas funciones, son parte del núcleo de PHP.

Configuración en tiempo de ejecución

El comportamiento de estas funciones está afectado por los valores definidos en `php.ini`.

Tabla 1. Opciones de Configuración de Variables

Nombre	Por defecto	Modificable
unserialize_callback_func	""	PHP_INI_ALL

Para más detalles y la definición de las constantes PHP_INI_* vea **ini_set()**.

A continuación se presenta una corta explicación de las directivas de configuración.

unserialize_callback_func **string**

> La llamada de retorno **unserialize()** será llamada (con el nombre de la clase no definida como parámetro), si el proceso de revertir la seriación encuentra una clase no definida que debería ser instanciada. Una advertencia aparecerá si la función especificada no está definida, o si la función no incluye o implementa la clase faltante. Así que únicamente defina esta entrada si de veras desea implementar tal función de retorno.
>
> Vea también **unserialize()**.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Esta extensión no tiene ninguna constante definida.

Tabla de contenidos

debug_zval_dump

(PHP 4 >= 4.2.0, PHP 5)

debug_zval_dump -- Dumps a string representation of an internal zend value to output

Descripción

void **debug_zval_dump** (mixed variable)

Dumps a string representation of an internal zend value to output.

Lista de parámetros

variable

The variable being evaluated.

Valores retornados

No value is returned.

Ejemplos

Ejemplo 1. debug_zval_dump() example

```
<?php
$var1 = 'Hello World';
$var2 = '';

$var2 =& $var1;

debug_zval_dump($var1);
?>
```

El resultado del ejemplo seria:

```
string(11) "Hello World" refcount(1)
```

Ver también

var_dump()
debug_backtrace()
References Explained

doubleval

(PHP 3, PHP 4 , PHP 5)

doubleval -- Obtiene el valor double (decimal) de una variable.

Descripción

double **doubleval** (mixed var)

Devuelve el valor double (decimal en punto flotante) de *var*.

var puede ser cualquier tipo escalar. No se puede usar **doubleval()** sobre arrays u objetos.

Ver también **intval()**, **strval()**, **settype()** y Type juggling.

empty

(PHP 3, PHP 4, PHP 5)

empty -- Determina si una variable está definida

Descripción

int **empty** (mixed var)

Devuelve **FALSE** si *var* está definida y tiene un valor no-vacío o distinto de cero; en otro caso devuelve **TRUE**.

Ver también **isset()** y **unset()**.

floatval

(PHP 4 >= 4.2.0, PHP 5)

floatval -- Obtener el valor flotante de una variable

Descripción

float **floatval** (mixed var)

Devuelve el valor **float** de *var*.

var puede ser cualquier tipo escalar. No puede usar **floatval()** en matrices u objetos.

```
<?php
$var = '122.34343El';
$valor_flotante_de_var = floatval($var);
echo $valor_flotante_de_var; // imprime 122.34343
?>
```

Vea también **intval()**, **strval()**, **settype()** y Conversión de Tipos.

get_defined_vars

(PHP 4 >= 4.0.4, PHP 5)

get_defined_vars -- Devuelve una matriz con todas la variables definidas

Descripción

array **get_defined_vars** (void)

Esta función devuelve una matriz multidimensional que contiene una lista de todas las variables definidas, ya sean variables de entorno, de servidor, o definidas por el usuario, al interior del contexto en el que **get_defined_vars()** es llamado. A partir de PHP 5, la variable $GLOBALS se incluye en los resultados de la matriz devuelta.

```
<?php
$b = array(1, 1, 2, 3, 5, 8);

$matriz = get_defined_vars();

// imprimir $b
print_r($matriz["b"]);

/* imprimir ruta al interpretre PHP (si es usado como CGI)
 * p.ej. /usr/local/bin/php */
echo $matriz["_"];

// imprimir los parametros de la linea de comandos, si existen
print_r($matriz["argv"]);

// imprimir todas las variables del servidor
print_r($matriz["_SERVER"]);

// imprimir todas las claves disponibles para las matrices de variables
print_r(array_keys(get_defined_vars()));
?>
```

Vea también **isset()**, **get_defined_functions()** y **get_defined_constants()**.

get_resource_type

(PHP 4 >= 4.0.2, PHP 5)

get_resource_type -- Devuelve el tipo de recurso

Descripción

string **get_resource_type** (resource gestor)

Esta función devuelve una cadena que representa el tipo del **resource** pasado a ella. Si el parámetro no es un **resource** válido, genera un error.

```
<?php
// imprime: mysql link
$c = mysql_connect();
echo get_resource_type($c) . "\n";

// imprime: file
$da = fopen("foo", "w");
echo get_resource_type($da) . "\n";

// imprime: domxml document
$doc = new_xmldoc("1.0");
echo get_resource_type($doc->doc) . "\n";
?>
```

gettype

(PHP 3, PHP 4 , PHP 5)

gettype -- Obtiene el tipo de una variable.

Descripción

string **gettype** (mixed var)

Devuelve el tipo de la variable PHP *var*.

Los valores posibles de la cadena devuelta son:

- "integer"
- "double"
- "string"
- "array"
- "object"
- "unknown type"

Ver también **settype()**.

import_request_variables

(PHP 4 >= 4.1.0, PHP 5)

import_request_variables -- Importar variables GET/POST/Cookie en el contexto global

Descripción

bool **import_request_variables** (string tipos [, string prefijo])

Importa las variables GET/POST/Cookie en el contexto global. Es útil si usted ha deshabilitado register_globals, pero desea ver algunas variables en el contexto global.

Usando el parámetro *tipos*, es posible indicar cuáles variables de petición deben importarse. Puede usar los caracteres 'G', 'P' y 'C' respectivamente para indicar GET, POST y Cookie. Estos caracteres no son sensibles a mayúsculas o minúsculas, así que puede usar cualquier combinación de 'g', 'p' y 'c'. POST incluye la información de archivos cargados mediante POST. Note que el orden de las letras tiene importancia, ya que cuando usa "gp", las variables POST sobrescribirán las variables GET con el mismo nombre. Cualquier otra letra diferente a GPC es descartada.

El parámetro *prefijo* es usado como prefijo para nombres de variables, que es colocado antes de todos los nombres de variables importados en el contexto global. De modo que si tiene un valor GET llamado "userid", y usa el prefijo "pref_", entonces obtendrá una variable global llamada $pref_userid.

Si está interesado en importar otras variables en el contexto global, como SERVER, considere el uso de **extract()**.

> **Nota:** Aunque el parámetro *prefijo* es opcional, recibirá un error de nivel E_NOTICE si no especifica el prefijo, o indica una cadena vacía como el prefijo. Este es un riesgo

potencial de seguridad. Los errores de nivel de noticia no son mostrados usando el nivel de reporte de errores predeterminado.

```
<?php
// Esto importara las variables GET y POST con el prefijo "rvar_"

import_request_variables("gP", "rvar_");

echo $rvar_foo;
?>
```

Vea también $_REQUEST, register_globals, Variables Predefinidas, y **extract()**.

intval

(PHP 3, PHP 4 , PHP 5)

intval -- Obtiene el valor entero de una variable.

Descripción

int **intval** (mixed var [, int base])

Devuelve el valor entero de *var*, usando la base de conversión especificada (por defecto es base 10).

var puede ser cualquier tipo escalar. No se puede usar **intval()** sobre arrays u objetos.

Ver también **doubleval()**, **strval()**, **settype()** y Type juggling.

is_array

(PHP 3, PHP 4 , PHP 5)

is_array -- Averigua si una variable es un array.

Descripción

int **is_array** (mixed var)

Devuelve **TRUE** si *var* es un array, y **FALSE** en otro caso.

Ver también **is_double()**, **is_float()**, **is_int()**, **is_integer()**, **is_real()**, **is_string()**, **is_long()**, y **is_object()**.

is_bool

(PHP 4 , PHP 5)

is_bool -- Encuentra si una variable es de tipo booleana

Descripción

bool **is_bool** (mixed var)

Devuelve **TRUE** si el parámetro *var* es un **boolean**.

Ejemplo 1. Ejemplos de is_bool()

```
<?php
$a = false;
$b = 0;

// Dado que $a es un booleano, esto es verdadero
if (is_bool($a)) {
    echo "Si, este es un booleano";
}

// Dado que $b no es un booleano, esto no es verdadero
if (is_bool($b)) {
    echo "Si, este es un booleano";
}
?>
```

Vea también **is_array()**, **is_float()**, **is_int()**, **is_integer()**, **is_string()**, y **is_object()**.

is_callable

(PHP 4 >= 4.0.6, PHP 5)

is_callable -- Verifica que los contenidos de una variable puedan ser llamados como una función

Descripción

bool **is_callable** (mixed var [, bool solo_sintaxis [, string &nombre_a_llamar]])

Verifica que los contenidos de una variable puedan ser llamados como una función. Esto permite revisar que los contenidos de una variable contengan el nombre de una función válida, o que una matriz contenga un objeto adecuadamente codificado y un nombre de función.

El parámetro *var* puede ser o bien el nombre de una función almacenada en una variable tipo cadena, o un objeto y el nombre de un método dentro del objeto, de este modo:

```
array($AlgunObjeto, 'NombreDelMetodo')
```

Si el argumento *solo_sintaxis* es **TRUE** la función solo verifica que *var* pueda ser una función o un método. Solo rechazará variables simples que no sean cadenas, o una matriz que no tenga una estructura válida para ser usada como llamada de retorno. Se espera que las matrices válidas tengan solo 2 entradas, la primera de las cuales es un objeto o una cadena, y la segunda una cadena.

El argumento *nombre_a_llamar* recibe el "nombre que puede ser llamado". En el ejemplo siguiente este es "algunaClase:algunMetodo". Note, sin embargo, que a pesar de la implicación de que algunaClase::algunMetodo() es un método estático que puede ser llamado, este no es el caso.

```php
<?php
//  Como chequear una variable para ver si puede ser llamada
//  como una funcion.

//
//  Variable simple que contiene una funcion
//

function algunaFuncion()
{
}

$variableFuncion = 'algunaFuncion';

var_dump(is_callable($variableFuncion, false, $nombre_a_llamar));  // bool(true)

echo $nombre_a_llamar, "\n";  // algunaFuncion

//
//  Matriz que contiene un metodo
//

class algunaClase {

  function algunMetodo()
  {
  }

}

$unObjeto = new algunaClase();

$variableMetodo = array($unObjeto, 'algunMetodo');

var_dump(is_callable($variableMetodo, true, $nombre_a_llamar));  //  bool(true)

echo $nombre_a_llamar, "\n";  //  algunaClase:algunMetodo

?>
```

is_double

(PHP 3, PHP 4 , PHP 5)

is_double -- Averigua si una variable es un valor double (número decimal).

Descripción

int **is_double** (mixed var)

Devuelve **TRUE** si *var* es un double (número decimal), y **FALSE** en otro caso.

Ver también **is_array()**, **is_float()**, **is_int()**, **is_integer()**, **is_real()**, **is_string()**, **is_long()**, y **is_object()**.

is_float

(PHP 3, PHP 4 , PHP 5)

is_float -- Averigua si una variable es un flotante.

Descripción

int **is_float** (mixed var)

Esta función es un alias de **is_double()**.

Ver también **is_double()**, **is_real()**, **is_int()**, **is_integer()**, **is_string()**, **is_object()**, **is_array()**, y **is_long()**.

is_int

(PHP 3, PHP 4 , PHP 5)

is_int -- Averigua si una variable es un valor entero.

Descripción

int **is_int** (mixed var)

Esta función es un alias de **is_long()**.

Ver también **is_double()**, **is_float()**, **is_integer()**, **is_string()**, **is_real()**, **is_object()**, **is_array()**, y **is_long()**.

is_integer

(PHP 3, PHP 4 , PHP 5)

is_integer -- Averigua si una variable es un valor entero.

Descripción

int **is_integer** (mixed var)

Esta función es un alias de **is_long()**.

Ver también **is_double()**, **is_float()**, **is_int()**, **is_string()**, **is_real()**, **is_object()**, **is_array()**, y **is_long()**.

is_long

(PHP 3, PHP 4 , PHP 5)

is_long -- Averigua si una varible es un valor entero.

Descripción

int **is_long** (mixed var)

Devuelve **TRUE** si *var* es un entero (long), y **FALSE** en otro caso.

Ver también is_double(), is_float(), is_int(), is_real(), is_string(), is_object(), is_array(), y is_integer().

is_null

(PHP 4 >= 4.0.4, PHP 5)

is_null -- Encuentra si una variable es **NULL**

Descripción

bool **is_null** (mixed var)

Devuelve **TRUE** si *var* es null, o **FALSE** en cualquier otro caso.

Consulte sobre el tipo **NULL** para una explicación de cuándo una variable es considerada como **NULL** y cuándo no.

Vea también **NULL**, is_bool(), is_numeric(), is_float(), is_int(), is_string(), is_object(), is_array(), is_integer(), y is_real().

is_numeric

(PHP 4 , PHP 5)

is_numeric -- Encuentra si una variable es un número o una cadena numérica

Descripción

bool **is_numeric** (mixed var)

Devuelve **TRUE** si *var* es un número o una cadena numérica, o **FALSE** en cualquier otro caso.

Vea también is_bool(), is_float(), is_int(), is_string(), is_object(), is_array(), y is_integer().

is_object

(PHP 3, PHP 4 , PHP 5)

is_object -- Averigua si una varible es un objeto.

Descripción

int **is_object** (mixed var)

Devuelve **TRUE** si *var* es un objeto, y **FALSE** en otro caso.

Ver también **is_long()**, **is_int()**, **is_integer()**, **is_float()**, **is_double()**, **is_real()**, **is_string()**, y **is_array()**.

is_real

(PHP 3, PHP 4 , PHP 5)

is_real -- Averigua si una varible es un número real.

Descripción

int **is_real** (mixed var)

Esta función es un alias de **is_double()**.

Ver también **is_long()**, **is_int()**, **is_integer()**, **is_float()**, **is_double()**, **is_object()**, **is_string()**, y **is_array()**.

is_resource

(PHP 4 , PHP 5)

is_resource -- Encuentra si una variable es un recurso

Descripción

bool **is_resource** (mixed var)

is_resource() devuelve **TRUE** si la variable dada por el parámetro *var* es un **resource**, o de lo contrario devuelve **FALSE**.

Ejemplo 1. uso de is_resource()

```
<?php

$enlace_bd = @mysql_connect('localhost', 'mysql_user', 'mysql_pass');
if (!is_resource($enlace_bd)) {
    die('No pudo realizarse la conexi&oacute;n : ' . mysql_error());
}

?>
```

Vea la documentación sobre el tipo de datos **resource** para más información.

is_scalar

(PHP 4 >= 4.0.5, PHP 5)

is_scalar -- Encuentra si una variable es un escalar

Descripción

bool **is_scalar** (mixed var)

is_scalar() devuelve **TRUE** si la variable dada por el parámetro *var* es un escalar, o de lo contrario devuelve **FALSE**.

Las variables escalares son aquellas que contienen un **integer**, **float**, **string** o **boolean**. Los tipos **array**, **object** y **resource** no son escalares.

```
<?php
function mostrar_var($var)
{
    if (is_scalar($var)) {
        echo $var;
    } else {
        var_dump($var);
    }
}
$pi = 3.1416;
$proteinas = array("hemoglobina", "citocromo c oxidasa", "ferredoxin");

mostrar_var($pi);
// imprime: 3.1416

mostrar_var($proteinas)
// imprime:
// array(3) {
//   [0]=>
//   string(10) "hemoglobina"
//   [1]=>
//   string(20) "citocromo c oxidasa"
//   [2]=>
//   string(10) "ferredoxin"
// }
?>
```

> **Nota: is_scalar()** no considera los valores de tipo **resource** como escalares dado que los recursos son tipos de datos abstractos que por el momento están basados en enteros. Este detalle de implementación no debería ser considerado confiable, ya que puede cambiar.

Vea también **is_bool()**, **is_numeric()**, **is_float()**, **is_int()**, **is_real()**, **is_string()**, **is_object()**, **is_array()**, y **is_integer()**.

is_string

(PHP 3, PHP 4 , PHP 5)

is_string -- Averigua si una varible es una cadena de caracteres (string).

Descripción

int **is_string** (mixed var)

Devuelve **TRUE** si *var* es una cadena, y **FALSE** en otro caso.

Ver también **is_long()**, **is_int()**, **is_integer()**, **is_float()**, **is_double()**, **is_real()**, **is_object()**, y **is_array()**.

isset

(PHP 3, PHP 4, PHP 5)

isset -- Determina si una varible está definida

Descripción

int **isset** (mixed var)

Devuelve **TRUE** si *var* existe; y **FALSE** en otro caso.

Si una variable ha sido destruida con **unset()**, ya no estará definida (no será **isset()**).

```
$a = "test";
echo isset($a); // true
unset($a);
echo isset($a); // false
```

Ver también **empty()** y **unset()**.

print_r

(PHP 4 , PHP 5)

print_r -- Imprime información legible para humanos sobre una variable

Descripción

bool **print_r** (mixed expresion [, bool devolver])

> **Nota:** El parámetro *devolver* fue agregado en PHP 4.3.0

print_r() despliega información sobre una variable en una forma que es apta para su lectura por humanos. Si se le entrega una variable tipo **string**, **integer** o **float**, el valor mismo será impreso. Si se le entrega un **array**, los valores serán presentados en un formato que muestra las claves y los elementos. Una notación parecida es usada para variables tipo **object**. **print_r()** y **var_export()** mostrarán también propiedades protegidas y privadas de objetos con PHP 5, en contraste con **var_dump()**.

Recuerde que **print_r()** desplazará el apuntador de la matriz al final. Use **reset()** para llevarlo de vuelta al comienzo.

```
<pre>
<?php
    $a = array ('a' => 'manzana', 'b' => 'banano', 'c' => array ('x', 'y', 'z'));
    print_r ($a);
?>
</pre>
```

Lo cual producirá la salida:

```
<pre>
Array
(
    [a] => manzana
    [b] => banano
    [c] => Array
        (
            [0] => x
            [1] => y
            [2] => z
        )
)
</pre>
```

Si quisiera capturar la salida de **print_r()**, use el parámetro *devolver*. Si este parámetro recibe el valor **TRUE**, **print_r()** devolverá su salida, en lugar de imprimirla (cosa que hace por defecto).

Ejemplo 1. Ejemplo del parámetro *devolver*

```
<?php
    $b = array ('m' => 'mono', 'foo' => 'bar', 'x' => array ('x', 'y', 'z'));
    $resultados = print_r($b, true);  // $resultados contiene ahora la
                                          salida de print_r
?>
```

> **Nota:** Si necesita capturar la salida de **print_r()** con una versión de PHP anterior a 4.3.0, use las funciones de control de salida.
>
> **Nota:** Antes de PHP 4.0.4, **print_r()** continuará ejecutándose indefinidamente si se le entrega una variable tipo **array** u **object** que contenga una referencia directa o indirecta a sí misma. Un ejemplo es *print_r($GLOBALS)* ya que la variable *$GLOBALS* es una variable global que contiene una referencia a sí misma.

Vea también **ob_start()**, **var_dump()** y **var_export()**.

serialize

(PHP 3>= 3.0.5, PHP 4 , PHP 5)

serialize -- Genera una representación apta para almacenamiento de un valor

Descripción

string **serialize** (mixed valor)

serialize() devuelve una cadena que contiene una representación de flujo de bytes del *valor* que puede ser almacenada en cualquier parte.

Esto es útil para el almacenamiento de valores en PHP sin perder su tipo y estructura.

Para recuperar el valor PHP a partir de la cadena seriada, use **unserialize()**. **serialize()** maneja todos los tipos, excepto **resource**. Usted puede incluso usar **serialize()** sobre matrices que contienen referencias a ellas mismas. Las referencias encontradas en la matriz/objeto que procede a seriar con **serialize()** también serán almacenadas.

Cuando seria objetos, PHP intentará llamar la función miembro **__sleep()** antes de la seriación. Esto permite que el objeto efectúe limpiezas de último minuto, etc. antes de ser seriado. De forma semejante, cuando un objeto es recuperado usando **unserialize()**, la función miembro **__wakeup()**

es llamada.

> **Nota:** En PHP 3, las propiedades de los objetos serán seriados, pero los métodos se pierden. Esa limitación fue retirada en PHP 4 ya que tanto las propiedades como los métodos son almacenados ahora. Por favor consulte la sección Seriación de Objetos de Clases y Objetos para más información.

Ejemplo 1. Ejemplo de serialize()

```
<?php
// $datos_sesion contiene una matriz multi-dimensional con
// informacion del usuario actual. Usamos serialize() para
// almacenarla en una base de datos al final de la peticion.


$con  = odbc_connect("bd_web", "php", "gallina");
$sent = odbc_prepare($con,
      "UPDATE sesiones SET datos = ? WHERE id = ?");
$datos_sql = array (serialize($datos_sesion), $PHP_AUTH_USER);

if (!odbc_execute($sent, &$datos_sql)) {
    $sent = odbc_prepare($con,
     "INSERT INTO sesiones (id, datos) VALUES(?, ?)");
    if (!odbc_execute($sent, &$datos_sql)) {
        /* Algo ha fallado.. */
    }
}
?>
```

Vea También: **unserialize()**.

settype

(PHP 3, PHP 4 , PHP 5)

settype -- Establece el tipo de una variable.

Descripción

int **settype** (string var, string type)

Establece el tipo de la varible *var* como *type*.

Los valores posibles para *type* son:

- "integer"
- "double"
- "string"
- "array"
- "object"

Devuelve **TRUE** si se lleva a cabo con éxito; en otro caso devuelve **FALSE**.

Ver también **gettype()**.

strval

(PHP 3, PHP 4 , PHP 5)

strval -- Obtiene una cadena de caracteres a partir de una variable.

Descripción

string **strval** (mixed var)

Devuelve una cadena con el valor de *var*.

var puede ser cualquier tipo escalar. No se puede usar **strval()** sobre arrays u objetos.

Ver también **doubleval()**, **intval()**, **settype()** y Type juggling.

unserialize

(PHP 3>= 3.0.5, PHP 4 , PHP 5)

unserialize -- Crea un valor PHP a partir de una representación almacenada

Descripción

mixed **unserialize** (string cadena)

unserialize() toma una variable sencilla seriada (vea **serialize()**) y la convierte de vuelta a su valor PHP. El valor convertido es retornado, y puede ser un **integer**, **float**, **string**, **array** u **object**. En caso de que la cadena pasada no pueda ser procesada para revertir la seriación, se devuelve **FALSE**.

> **Directiva unserialize_callback_func:** Es posible establecer una función-llamada de retorno la cual será llamada si una clase no definida debería ser instanciada durante el proceso de revertir la seriación. (para prevenir que se reciba un **object** incompleto "__PHP_Incomplete_Class".) Use su `php.ini`, **ini_set()** o `.htaccess` para definir 'unserialize_callback_func'. Cada vez que una clase no definida deba ser instanciada, ésta función será llamada. Para deshabilitar esta característica simplemente asigne un valor vacío a este parámetro. También note que la directiva unserialize_callback_func se hizo disponible en PHP 4.2.0.

Si la variable que está siendo convertida de vuelta es un objeto, PHP intentará llamar la función miembro **__wakeup()** (si existe) automáticamente luego de haber recontruido satisfactoriamente el objeto.

Ejemplo 1. Ejemplo de unserialize_callback_func

```
<?php
$objeto_seriado='O:1:"a":1:{s:5:"valor";s:3:"100";}';

// la directiva unserialize_callback_func esta disponible a partir de PHP 4.2.0
ini_set('unserialize_callback_func', 'mi_llamada_de_retorno'); // defina su callback_fu

function mi_llamada_de_retorno($nombre_clase)
{
    // tan solo incluya un archivo que contenga su definicion de clase

    // usted recibe $nombre_clase para determinar que definicion de
    // clase requiere
}
?>
```

Nota: En PHP 3, los métodos no se preservan cuando se revierte un objeto seriado. Esa limitación fue retirada en PHP 4 ya que tanto las propiedades como los métodos se almacenan ahora. Por favor consulte la sección Seriación de Objetos de Clases y Objetos para más información.

Ejemplo 2. Ejemplo de unserialize()

```
<?php
// Aqui usamos unserialize() para cargar los datos de sesion
// provenientes de la cadena seleccionada desde la base de datos en la
// matriz $datos_sesion. Este ejemplo complementa aquel descrito con
// serialize().

$con  = odbc_connect("bd_web", "php", "gallina");
$sent = odbc_prepare($con, "SELECT datos FROM sesiones WHERE id = ?");
$datos_sql = array ($PHP_AUTH_USER);

if (!odbc_execute($sent, &$datos_sql) || !odbc_fetch_into($sent, &$tmp)) {
    // si la ejecucion del comando o la recuperacion de datos falla,
    // inicializar una matriz vacia
    $datos_sesion = array();
} else {
    // ahora deberiamos tener los datos seriados en $tmp[0].
    $datos_sesion = unserialize($tmp[0]);
    if (!is_array($datos_sesion)) {
        // algo ha fallado, inicializar una matriz vacia
        $datos_sesion = array();
    }
}
?>
```

Vea también **serialize()**.

unset

(PHP 3, PHP 4, PHP 5)

unset -- Destruye una variable dada

Descripción

int **unset** (mixed var)

unset() destruye la variable especificada y devuelve **TRUE**.

Ejemplo 1. Ejemplo de unset()

```
unset( $foo );
unset( $bar['quux'] );
```

Ver también **isset()** y **empty()**.

var_dump

(PHP 3>= 3.0.5, PHP 4 , PHP 5)

var_dump -- Vuelca información sobre una variable

Descripción

void **var_dump** (mixed expresion [, mixed expresion [, ...]])

Esta función despliega información estructurada sobre una o más expresiones que incluye sus tipos y valores. Las matrices y los objetos son exploradas recursivamente con valores sangrados para mostrar su estructura.

En PHP sólo las propiedades públicas de los objetos serán devueltas en la salida. **var_export()** y **print_r()** devolverán también las propiedadse protegidas y privadas.

> **Sugerencia:** Como con todo lo que presenta un resultado directamente en el navegador, se pueden utilizar las funciones de control de salida para capturar el resultado de esta función y grabarlo - por ejemplo - en una **string**.

Ejemplo 1. Ejemplo de var_dump()

```
<?php
$a = array (1, 2, array ("a", "b", "c"));
var_dump ($a);

?>
```

Salida:

```
array(3) {
  [0]=>
  int(1)
  [1]=>
  int(2)
  [2]=>
  array(3) {
    [0]=>
    string(1) "a"
    [1]=>
    string(1) "b"
    [2]=>
    string(1) "c"
  }
}

<?php

$b = 3.1;
$c = true;
var_dump($b, $c);

?>
```

salida:

```
float(3.1)
bool(true)
```

Vea también **var_export()** y **print_r()**.

var_export

(PHP 4 >= 4.2.0, PHP 5)

var_export -- Imprime o devuelve una representación de cadena, apta para su procesamiento, de una variable

Descripción

mixed **var_export** (mixed expresion [, bool devolver])

Esta función devuelve información estructurada sobre la variable que le es pasada. Es similar a **var_dump()** con dos excepciones. La primera es que la representación devuelta es código PHP válido, la segunda, que también devolverá propiedades protegidas y privadas de un objeto con PHP 5.

También puede devolver la representación de la variable usando **TRUE** como segundo parámetro para esta función.

```
<?php
$a = array (1, 2, array ("a", "b", "c"));
var_export ($a);
?>
```

salida:

```
array (
  0 => 1,
  1 => 2,
  2 =>
  array (
    0 => 'a',
    1 => 'b',
    2 => 'c',
  ),
)
```

```
<?php

$b = 3.1;
$v = var_export($b, true);
echo $v;

?>
```

salida:

```
3.1
```

Vea también **var_dump()** y **print_r()**.

CXXX. vpopmail Functions

Introducción

Aviso

Esta extensión es *EXPERIMENTAL*. Esto significa que el comportamiento de esta extensión, los nombre de sus funciones y en definitiva TODO lo documentado sobre esta extensión, puede cambiar en una futura versión de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

This extension has been moved from PHP as of PHP 4.3.0 and now vpopmail lives in PECL.

Instalación

As of PHP 4, these functions are only available if PHP was configured with *--with-vpopmail [=DIR]*.

Tabla de contenidos

vpopmail_add_alias_domain_ex

(4.0.5 - 4.2.3 only)

vpopmail_add_alias_domain_ex -- Add alias to an existing virtual domain

Description

bool **vpopmail_add_alias_domain_ex** (string olddomain, string newdomain)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

vpopmail_add_alias_domain

(4.0.5 - 4.2.3 only)

vpopmail_add_alias_domain -- Add an alias for a virtual domain

Description

bool **vpopmail_add_alias_domain** (string domain, string aliasdomain)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

vpopmail_add_domain_ex

(4.0.5 - 4.2.3 only)

vpopmail_add_domain_ex -- Add a new virtual domain

Description

bool **vpopmail_add_domain_ex** (string domain, string passwd [, string quota [, string bounce [, bool apop]]])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

vpopmail_add_domain

(4.0.5 - 4.2.3 only)

vpopmail_add_domain -- Add a new virtual domain

Description

bool **vpopmail_add_domain** (string domain, string dir, int uid, int gid)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

vpopmail_add_user

(4.0.5 - 4.2.3 only)

vpopmail_add_user -- Add a new user to the specified virtual domain

Description

bool **vpopmail_add_user** (string user, string domain, string password [, string gecos [, bool apop]])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

vpopmail_alias_add

(4.1.0 - 4.2.3 only)

vpopmail_alias_add -- Insert a virtual alias

Description

bool **vpopmail_alias_add** (string user, string domain, string alias)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

vpopmail_alias_del_domain

(4.1.0 - 4.2.3 only)

vpopmail_alias_del_domain -- Deletes all virtual aliases of a domain

Description

bool **vpopmail_alias_del_domain** (string domain)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

vpopmail_alias_del

(4.1.0 - 4.2.3 only)

vpopmail_alias_del -- Deletes all virtual aliases of a user

Description

bool **vpopmail_alias_del** (string user, string domain)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

vpopmail_alias_get_all

(4.1.0 - 4.2.3 only)

vpopmail_alias_get_all -- Get all lines of an alias for a domain

Description

array **vpopmail_alias_get_all** (string domain)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

vpopmail_alias_get

(4.1.0 - 4.2.3 only)

vpopmail_alias_get -- Get all lines of an alias for a domain

Description

array **vpopmail_alias_get** (string alias, string domain)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

vpopmail_auth_user

(4.0.5 - 4.2.3 only)

vpopmail_auth_user -- Attempt to validate a username/domain/password

Description

bool **vpopmail_auth_user** (string user, string domain, string password [, string apop])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

vpopmail_del_domain_ex

(4.0.5 - 4.2.3 only)

vpopmail_del_domain_ex -- Delete a virtual domain

Description

bool **vpopmail_del_domain_ex** (string domain)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

vpopmail_del_domain

(4.0.5 - 4.2.3 only)

vpopmail_del_domain -- Delete a virtual domain

Description

bool **vpopmail_del_domain** (string domain)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

vpopmail_del_user

(4.0.5 - 4.2.3 only)

vpopmail_del_user -- Delete a user from a virtual domain

Description

bool **vpopmail_del_user** (string user, string domain)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

vpopmail_error

(4.0.5 - 4.2.3 only)

vpopmail_error -- Get text message for last vpopmail error

Description

string **vpopmail_error** (void)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

vpopmail_passwd

(4.0.5 - 4.2.3 only)

vpopmail_passwd -- Change a virtual user's password

Description

bool **vpopmail_passwd** (string user, string domain, string password [, bool apop])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

vpopmail_set_user_quota

(4.0.5 - 4.2.3 only)

vpopmail_set_user_quota -- Sets a virtual user's quota

Description

bool **vpopmail_set_user_quota** (string user, string domain, string quota)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

CXXXI. W32api Functions

Introducción

This extension is a generic extension API to DLLs. This was originally written to allow access to the Win32 API from PHP, although you can also access other functions exported via other DLLs.

Currently supported types are generic PHP types (strings, booleans, floats, integers and nulls) and

types you define with **w32api_deftype()**.

> **Nota:** This extension has been moved to the PECL repository and is no longer bundled with PHP as of PHP 5.1.0.

Aviso
Esta extensión es *EXPERIMENTAL*. Esto significa que el comportamiento de esta extensión, los nombre de sus funciones y en definitiva TODO lo documentado sobre esta extensión, puede cambiar en una futura versión de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Requirimientos

This extension will only work on Windows systems.

Instalación

No se necesita ninguna instalación para usar estas funciones, son parte del núcleo de PHP.

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

This extension defines one resource type, used for user defined types. The name of this resource is *"dynaparm"*.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

DC_MICROSOFT (**integer**)

DC_BORLAND (**integer**)

DC_CALL_CDECL (**integer**)

DC_CALL_STD (**integer**)

DC_RETVAL_MATH4 (**integer**)

DC_RETVAL_MATH8 (integer)

DC_CALL_STD_BO (integer)

DC_CALL_STD_MS (integer)

DC_CALL_STD_M8 (integer)

DC_FLAG_ARGPTR (integer)

Ejemplos

This example gets the amount of time the system has been running and displays it in a message box.

Ejemplo 1. Get the uptime and display it in a message box

```
<?php
// Define constants needed, taken from
// Visual Studio/Tools/Winapi/WIN32API.txt
define("MB_OK", 0);

// Load the extension in
dl("php_w32api.dll");

// Register the GetTickCount function from kernel32.dll
w32api_register_function("kernel32.dll",
                         "GetTickCount",
                         "long");

// Register the MessageBoxA function from User32.dll
w32api_register_function("User32.dll",
                         "MessageBoxA",
                         "long");

// Get uptime information
$ticks = GetTickCount();

// Convert it to a nicely displayable text
$secs  = floor($ticks / 1000);
$mins  = floor($secs / 60);
$hours = floor($mins / 60);

$str = sprintf("You have been using your computer for:" .
               "\r\n %d Milliseconds, or \r\n %d Seconds" .
               "or \r\n %d mins or\r\n %d hours %d mins.",
               $ticks,
               $secs,
               $mins,
               $hours,
               $mins - ($hours*60));

// Display a message box with only an OK button and the uptime text
MessageBoxA(NULL,
            $str,
            "Uptime Information",
            MB_OK);
?>
```

Tabla de contenidos

name
w32api_register_function -- Registers function function_name from library with PHP
w32api_set_call_method -- Sets the calling method used

w32api_deftype

(4.2.0 - 4.2.3 only)

w32api_deftype -- Defines a type for use with other w32api_functions

Description

bool **w32api_deftype** (string typename, string member1_type, string member1_name [, string ... [, string ...]])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

If you would like to define a type for a w32api call, you need to call **w32api_deftype()**. This function takes 2n+1 arguments, where n is the number of members the type has. The first argument is the name of the type. After that is the type of the member followed by the members name (in pairs). A member type can be a user defined type. All the type names are case sensitive. Built in type names should be provided in lowercase. Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

w32api_init_dtype

(4.2.0 - 4.2.3 only)

w32api_init_dtype -- Creates an instance of the data type typename and fills it with the values passed

Description

resource **w32api_init_dtype** (string typename, mixed value [, mixed ...])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

This function creates an instance of the data type named *typename*, filling in the values of the data type. The *typename* parameter is case sensitive. You should give the values in the same order as you defined the data type with **w32api_deftype()**. The type of the resource returned is *dynaparm*.

w32api_invoke_function

(4.2.0 - 4.2.3 only)

w32api_invoke_function -- Invokes function funcname with the arguments passed after the function name

Description

mixed **w32api_invoke_function** (string funcname, mixed argument [, mixed ...])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

w32api_invoke_function() tries to find the previously registered function, named *funcname*, passing the parameters you provided. The return type is the one you set when you registered the function, the value is the one returned by the function itself. Any of the arguments can be of any PHP type or **w32api_deftype()** defined type, as needed.

w32api_register_function

(4.2.0 - 4.2.3 only)

w32api_register_function -- Registers function function_name from library with PHP

Description

bool **w32api_register_function** (string library, string function_name, string return_type)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

This function tries to find the *function_name* function in *libary*, and tries to import it into PHP. The function will be registered with the given *return_type*. This type can be a generic PHP type, or a type defined with **w32api_deftype()**. All type names are case sensitive. Built in type names should be provided in lowercase. Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

w32api_set_call_method

(4.2.0 - 4.2.3 only)

w32api_set_call_method -- Sets the calling method used

Description

void **w32api_set_call_method** (int method)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

This function sets the method call type. The parameter can be one of the constants **DC_CALL_CDECL** or **DC_CALL_STD**. The extension default is **DC_CALL_STD**.

CXXXII. Funciones WDDX

Estas funciones permiten el uso de WDDX.

Debe saber que todas las funciones que serializan variables usan el primer elemento de un array para determinar si este ha de serializarse en forma de array o como estructura. Si el primer elemento esta indexado por una cadena, se serializa como estructura, y en caso contrario, como array.

Ejemplo 1. Serializacion de un valor simple

```
<?php
print wddx_serialize_value("Ejemplo de PHP a paquete WDDX", "Paquete PHP");
?>
```

Este ejemplo producira:

```
<wddxPacket version='0.9'><header comment='Paquete PHP'/><data>
<string>Ejemplo de PHP a paquete WDDX</string></data></wddxPacket>
```

Ejemplo 2. Uso de paquetes incrementales

```
<?php
$pi = 3.1415926;
$packet_id = wddx_packet_start("PHP");
wddx_add_vars($packet_id, "pi");

/* Suponga que $ciudades se ha obtenido de una base de datos */
$ciudades = array("Austin", "Novato", "Seattle");
wddx_add_vars($packet_id, "ciudades");

$packet = wddx_packet_end($packet_id);
print $packet;
?>
```

Este ejemplo producira:

```
<wddxPacket version='0.9'><header comment='PHP'/><data><struct>
<var name='pi'><number>3.1415926</number></var><var name='ciudades'>
<array length='3'><string>Austin</string><string>Novato</string>
<string>Seattle</string></array></var></struct></data></wddxPacket>
```

Tabla de contenidos

wddx_add_vars

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

wddx_add_vars -- Finaliza un paquete WDDX con el identificador dado

Descripcion

wddx_add_vars (entero packet_id, varios-tipos name_var [, varios-tipos ...])

wddx_add_vars() se utiliza para serializar las variables dadas e incorporar el resultado al paquete especificado por *packet_id*. Las variables a serializar se especifican exactamente igual que en **wddx_serialize_vars()**.

wddx_deserialize

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

wddx_deserialize -- Des-serializa un paquete WDDX

Descripcion

varios-tipos **wddx_deserialize** (cadena packet)

wddx_deserialized() toma una cadena *packet* y la desserializa. Devuelve el resultado que puede ser de tipo cadena, numerico o array. Las estructuras son desserializadas en forma de arrays asociativos.

wddx_packet_end

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

wddx_packet_end -- Finaliza un paquete WDDX con el identificador dado

Descripcion

cadena **wddx_packet_end** (entero packet_id)

wddx_packet_end() finaliza el paquete WDDX especificado por el *packet_id* y devuelve la cadena con el paquete.

wddx_packet_start

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

wddx_packet_start -- Comienza un nuevo paquete WDDX con una estructura dentro

Descripcion

entero **wddx_packet_start** ([cadena comentario])

Utilice **wddx_packet_start()** para comenzar un nuevo paquete WDDX que permita la adicion sucesiva de variables. Recibe el parametro opcional *comentario* y devuelve un identificador de paquete para su uso en posteriores llamadas. Automaticamente define una estructura dentro del paquete para contener las variables.

wddx_serialize_value

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

wddx_serialize_value -- Serializa un valor simple en un paquete WDDX

Descripcion

cadena **wddx_serialize_value** (varios-tipos var [, cadena comentario])

wddx_serialize_value() se utiliza para crear un paquete WDDX desde un valor simple dado. Toma el valor contenido en *var*, y una cadena *comentario* opcional que aparecera en la cabecera del paquete, y devuelve el paquete WDDX.

wddx_serialize_vars

(PHP 3>= 3.0.7, PHP 4 , PHP 5)

wddx_serialize_vars -- Serializa variables en un paquete WDDX

Descripcion

cadena **wddx_serialize_vars** (varios-tipos nombre_var [, varios-tipos ...])

wddx_serialize_vars() se utiliza para crear un paquete WDDX con una estructura que contiene la representacion serializada de las variables pasadas como parametros.

wddx_serialize_vars() toma un numero variable de argumentos, cada uno de los cuales puede ser una cadena con el nombre de una variable o un array con nombres de variables, o de otro array, etc.

Ejemplo 1. wddx_serialize_vars example

```
<?php
$a = 1;
$b = 5.5;
$c = array("azul", "naranja", "violeta");
$d = "colores";

$clvars = array("c", "d");
print wddx_serialize_vars("a", "b", $clvars);
?>
```

El ejemplo anterior producira:

```
<wddxPacket version='0.9'><header/><data><struct><var name='a'><number>1</number></var>
<var name='b'><number>5.5</number></var><var name='c'><array length='3'>
<string>azul</string><string>naranja</string><string>violeta</string></array></var>
<var name='d'><string>colores</string></var></struct></data></wddxPacket>
```

CXXXIII. xattr Functions

Introducción

The xattr extension allows for the manipulation of extended attributes on a filesystem.

Requirimientos

To use xattr, you will need libattr installed. It is available at http://oss.sgi.com/projects/xfs/.

> **Nota:** These functions only work on filesystems that support extended attributes, and have them enabled at mount time. Some common filesystems that support extended attributes are ext2, ext3, reiserfs, jfs, and xfs.

Instalación

xattr is currently available through PECL http://pecl.php.net/package/xattr.

If PEAR is available on your *nix-like system you can use the pear installer to install the xattr extension, by the following command: **pear -v install xattr**.

You can always download the tar.gz package and install xattr by hand:
Ejemplo 1. xattr install by hand

```
gunzip xattr-xxx.tgz
tar -xvf xattr-xxx.tar
cd xattr-xxx
phpize
./configure && make && make install
```

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

XATTR_ROOT (integer)

Set attribute in root (trusted) namespace. Requires root privileges.

XATTR_DONTFOLLOW (integer)

Do not follow the symbolic link but operate on symbolic link itself.

XATTR_CREATE (integer)

Function will fail if extended attribute already exists.

XATTR_REPLACE (integer)

Function will fail if extended attribute doesn't exist.

Tabla de contenidos

xattr_get

(no version information, might be only in CVS)

xattr_get -- Get an extended attribute

Descripción

string **xattr_get** (string filename, string name [, int flags])

This function gets the value of an extended attribute of a file.

Extended attributes have two different namespaces: user and root namespace. User namespace is available for all users while root namespace is available only for user with root privileges. xattr operates on user namespace by default, but you can change that using *flags* argument.

Lista de parámetros

filename

The file from which we get the attribute.

name

The name of the attribute.

flags

Tabla 1. Supported xattr flags

XATTR_DONTFOLLOW	Do not follow the symbolic link but operate on symbolic link itself.
XATTR_ROOT	Set attribute in root (trusted) namespace. Requires root privileges.

Valores retornados

Returns a string containing the value or **FALSE** if the attribute doesn't exist.

Ejemplos

Ejemplo 1. Checks if system administrator has signed the file

```
<?php
$file = '/usr/local/sbin/some_binary';
$signature = xattr_get($file, 'Root signature', XATTR_ROOT);

/* ... check if $signature is valid ... */

?>
```

Ver también

xattr_list()
xattr_set()
xattr_remove()

xattr_list

(no version information, might be only in CVS)

xattr_list -- Get a list of extended attributes

Descripción

array **xattr_list** (string filename [, int flags])

This functions gets a list of names of extended attributes of a file.

Extended attributes have two different namespaces: user and root namespace. User namespace is available for all users while root namespace is available only for user with root privileges. xattr operates on user namespace by default, but you can change that using *flags* argument.

Lista de parámetros

filename

The path of the file.

flags

Tabla 1. Supported xattr flags

XATTR_DONTFOLLOW	Do not follow the symbolic link but operate on symbolic link itself.

XATTR_ROOT	Set attribute in root (trusted) namespace. Requires root privileges.

Valores retornados

This function returns an array with names of extended attributes.

Ejemplos

Ejemplo 1. Prints names of all extended attributes of file

```
<?php
$file = 'some_file';
$root_attributes = xattr_list($file, XATTR_ROOT);
$user_attributes = xattr_list($file);

echo "Root attributes: \n";
foreach ($root_attributes as $attr_name) {
    printf("%s\n", $attr_name);
}

echo "\n User attributes: \n";
foreach ($attributes as $attr_name) {
    printf("%s\n", $attr_name);
}

?>
```

Ver también

xattr_get()

xattr_remove

(no version information, might be only in CVS)

xattr_remove -- Remove an extended attribute

Descripción

bool **xattr_remove** (string filename, string name [, int flags])

This function removes an extended attribute of a file.

Extended attributes have two different namespaces: user and root namespace. User namespace is available for all users while root namespace is available only for user with root privileges. xattr operates on user namespace by default, but you can change that using *flags* argument.

Lista de parámetros

filename

The file from which we remove the attribute.

name

The name of the attribute to remove.

flags

Tabla 1. Supported xattr flags

XATTR_DONTFOLLOW	Do not follow the symbolic link but operate on symbolic link itself.
XATTR_ROOT	Set attribute in root (trusted) namespace. Requires root privileges.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplos

Ejemplo 1. Removes all extended attributes of a file

```
<?php
$file = 'some_file';
$attributes = xattr_list($file);

foreach ($attributes as $attr_name) {
    xattr_remove($file, $attr_name);
}
?>
```

Ver también

xattr_list()
xattr_set()
xattr_get()

xattr_set

(no version information, might be only in CVS)

xattr_set -- Set an extended attribute

Descripción

bool **xattr_set** (string filename, string name, string value [, int flags])

This function sets the value of an extended attribute of a file.

Extended attributes have two different namespaces: user and root namespace. User namespace is available for all users while root namespace is available only for user with root privileges. xattr operates on user namespace by default, but you can change that using *flags* argument.

Lista de parámetros

filename

The file in which we set the attribute.

name

The name of the extended attribute. This attribute will be created if it doesn't exist or replaced otherwise. You can change this behaviour by using the *flags* parameter.

value

The value of the attribute.

flags

Tabla 1. Supported xattr flags

XATTR_CREATE	Function will fail if extended attribute already exists.
XATTR_REPLACE	Function will fail if extended attribute doesn't exist.
XATTR_DONTFOLLOW	Do not follow the symbolic link but operate on symbolic link itself.
XATTR_ROOT	Set attribute in root (trusted) namespace. Requires root privileges.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplos

Ejemplo 1. Sets extended attributes on .wav file

```
<?php
$file = 'my_favourite_song.wav';
xattr_set($file, 'Artist', 'Someone');
xattr_set($file, 'My ranking', 'Good');
xattr_set($file, 'Listen count', '34');

/* ... other code ... */

printf("You've played this song %d times", xattr_get($file, 'Listen count'));
?>
```

Ver también

xattr_get()
xattr_remove()

xattr_supported

(no version information, might be only in CVS)

xattr_supported -- Check if filesystem supports extended attributes

Descripción

bool **xattr_supported** (string filename [, int flags])

This functions checks if the filesystem holding the given file supports extended attributes. Read access to the file is required.

Lista de parámetros

filename

The path of the tested file.

flags

Tabla 1. Supported xattr flags

XATTR_DONTFOLLOW	Do not follow the symbolic link but operate on symbolic link itself.

Valores retornados

This function returns **TRUE** if filesystem supports extended attributes, **FALSE** if it doesn't and **NULL** if it can't be determined (for example wrong path or lack of permissions to file).

Ejemplos

Ejemplo 1. xattr_supported() example

The following code checks if we can use extended attributes.

```
<?php
$file = 'some_file';

if (xattr_supported($file)) {
    /* ... make use of some xattr_* functions ... */
}

?>
```

Ver también

xattr_get()
xattr_list()

CXXXIV. xdiff Functions

Introducción

xdiff extension creates and applies patches to both text and binary files.

Requirimientos

To use xdiff, you will need libxdiff installed, available on the libxdiff homepage http://www.xmailserver.org/xdiff-lib.html.

> **Nota:** You'll need at least libxdiff 0.7 for these functions to be aware of memory_limit.

Instalación

xdiff is currently available through PECL http://pecl.php.net/package/xdiff.

If PEAR is available on your *nix-like system you can use the pear installer to install the xdiff extension, by the following command: **pear -v install xdiff**.

You can always download the tar.gz package and install xdiff by hand:
Ejemplo 1. xdiff install by hand

```
gunzip xdiff-xxx.tgz
tar -xvf xdiff-xxx.tar
cd xdiff-xxx
phpize
./configure && make && make install
```

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

`XDIFF_PATCH_NORMAL` (**integer**)

`XDIFF_PATCH_REVERSE` (**integer**)

Tabla de contenidos

xdiff_string_patch -- Patch a string with an unified diff

xdiff_file_diff_binary

(no version information, might be only in CVS)

xdiff_file_diff_binary -- Make binary diff of two files

Description

bool **xdiff_file_diff_binary** (string file1, string file2, string dest)

xdiff_file_diff_binary() makes binary diff of files *file1* and *file2* and stores result in file *dest*. This function works with both text and binary files. Resulting file is in binary format.

> **Nota:** Both files will be loaded into memory so ensure that your memory_limit is set high enough.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplo 1. xdiff_file_diff_binary() example

The following code makes binary diff of two archives.

```
<?php
$old_version = 'my_script_1.0.tgz';
$new_version = 'my_script_1.1.tgz';

xdiff_file_diff_binary($old_version, $new_version, 'my_script.bdiff');
?>
```

See also **xdiff_string_diff_binary()**.

xdiff_file_diff

(no version information, might be only in CVS)

xdiff_file_diff -- Make unified diff of two files

Description

bool **xdiff_file_diff** (string file1, string file2, string dest [, int context [, bool minimal]])

xdiff_file_diff() makes unified diff of files *file1* and *file2* and stores result in file *dest*. *context* indicated how many lines of context you want to include in diff result. Set *minimal* to **TRUE** if you want to minimalize size of diff (can take a long time). Resulting file is human-readable.

> **Nota:** This function doesn't work well with binary files. To make diff of binary files use **xdiff_file_diff_binary()**.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplo 1. xdiff_file_diff() example

The following code makes unified diff of two php files.

```
<?php
$old_version = 'my_script.php';
$new_version = 'my_new_script.php';

xdiff_file_diff($old_version, $new_version, 'my_script.diff', 2);
?>
```

See also **xdiff_string_diff()**.

xdiff_file_merge3

(no version information, might be only in CVS)

xdiff_file_merge3 -- Merge 3 files into one

Description

mixed **xdiff_file_merge3** (string file1, string file2, string file3, string dest)

xdiff_file_merge3() merges files *file1*, *file2* and *file3* into one and stores result in file *dest*.

Returns **TRUE** if merge was successful, string with rejected chunks if it was not or **FALSE** if an internal error happened.

Ejemplo 1. xdiff_file_merge3() example

The following code merges three files into one.

```
<?php
$old_version = 'original_script.php';
$fix1 = 'script_with_fix1.php';
$fix2 = 'script_with_fix2.php';

$errors = xdiff_file_merge3($old_version, $fix1, $fix2, 'fixed_script.php');
if (is_string($errors)) {
    echo "Rejects:\n";
    echo $errors;
}
?>
```

See also **xdiff_string_merge3()**.

xdiff_file_patch_binary

(no version information, might be only in CVS)

xdiff_file_patch_binary -- Patch a file with a binary diff

Description

bool **xdiff_file_patch_binary** (string file, string patch, string dest)

xdiff_file_patch_binary() patches file *file* with binary patch in file *patch* and stores result in file *dest*.

> **Nota:** Both files (file and patch) will be loaded into memory so ensure that your memory_limit is set high enough.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplo 1. xdiff_file_patch_binary() example

The following code applies binary diff to a file.

```
<?php
$old_version = 'archive-1.0.tgz';
$patch = 'archive.bpatch';

$result = xdiff_file_patch_binary($old_version, $patch, 'archive-1.1.tgz');
if ($result) {
   echo "File patched";
} else {
   echo "File couldn't be patched";
}

?>
```

See also **xdiff_string_patch_binary()**.

xdiff_file_patch

(no version information, might be only in CVS)

xdiff_file_patch -- Patch a file with an unified diff

Description

mixed **xdiff_file_patch** (string file, string patch, string dest [, int flags])

xdiff_file_patch() patches file *file* with unified patch in file *patch* and stores result in file *dest*.

flags can be either **XDIFF_PATCH_NORMAL** (default mode, normal patch) or **XDIFF_PATCH_REVERSE** (reversed patch).

Returns **FALSE** if an internal error happened, string with rejected chunks of patch or **TRUE** if patch has been successfully applied.

Ejemplo 1. xdiff_file_patch() example

The following code applies unified diff to a file.

```
<?php
$old_version = 'my_script-1.0.php';
$patch = 'my_script.patch';

$errors = xdiff_file_patch($old_version, $patch, 'my_script-1.1.php');
if (is_string($errors)) {
   echo "Rejects:\n";
   echo $errors;
}

?>
```

Ejemplo 2. Patch reversing example

The following code reverses a patch.

```
<?php
$new_version = 'my_script-1.1.php';
$patch = 'my_script.patch';

$errors = xdiff_file_patch($new_version, $patch, 'my_script-1.0.php', XDIFF_PATCH_REVER
if (is_string($errors)) {
   echo "Rejects:\n";
   echo $errors;
}

?>
```

See also **xdiff_string_patch()**.

xdiff_string_diff_binary

(no version information, might be only in CVS)

xdiff_string_diff_binary -- Make binary diff of two strings

Description

mixed **xdiff_string_diff_binary** (string str1, string str2)

xdiff_string_diff_binary() makes binary diff of strings *str1* and *str2*.

Returns string with result or **FALSE** if an internal error happened.

See also **xdiff_file_diff_binary()**.

xdiff_string_diff

(no version information, might be only in CVS)

xdiff_string_diff -- Make unified diff of two strings

Description

mixed **xdiff_string_diff** (string str1, string str2 [, int context [, bool minimal]])

xdiff_string_diff() makes unified diff of strings *str1* and *str2*. *context* indicated how many lines of context you want to include in diff result. Set *minimal* to **TRUE** if you want to minimalize size of diff (can take a long time).

> **Nota:** This function doesn't work well with binary strings. To make diff of binary strings use **xdiff_string_diff_binary()**.

Returns string with result or **FALSE** if an internal error happened.

Ejemplo 1. xdiff_string_diff() example

The following code makes unified diff of two articles.

```
<?php
$old_article = file_get_contents('./old_article.txt');
$new_article = $_REQUEST['article']; /* Let's say that someone pasted a new article to

$diff = xdiff_string_diff($old_article, $new_article, 1);
if (is_string($diff)) {
    echo "Differences between two articles:\n";
    echo $diff;
}

?>
```

See also **xdiff_file_diff()**.

xdiff_string_merge3

(no version information, might be only in CVS)

xdiff_string_merge3 -- Merge 3 strings into one

Description

string **xdiff_string_merge3** (string str1, string str2, string str3 [, string &error])

xdiff_string_merge3() merges strings *str1*, *str2* and *str3* into one.

If *error* is passed then rejected parts are stored inside this variable.

Returns merged string or **FALSE** if an internal error happened.

See also **xdiff_file_merge3()**.

xdiff_string_patch_binary

(no version information, might be only in CVS)

xdiff_string_patch_binary -- Patch a string with a binary diff

Description

string **xdiff_string_patch_binary** (string str, string patch)

xdiff_string_patch_binary() patches string *str* with binary patch in string *patch*.

Returns a patched string.

See also **xdiff_file_patch_binary()**.

xdiff_string_patch

(no version information, might be only in CVS)

xdiff_string_patch -- Patch a string with an unified diff

Description

string **xdiff_string_patch** (string str, string patch [, int flags [, string &error]])

xdiff_string_patch() patches string *str* with unified patch in string *patch*.

flags can be either **XDIFF_PATCH_NORMAL** (default mode, normal patch) or **XDIFF_PATCH_REVERSE** (reversed patch).

If *error* is passed then rejected parts are stored inside this variable.

Ejemplo 1. xdiff_string_patch() example

The following code applies changes to some article.

```
<?php
$old_article = file_get_contents('./old_article.txt');
$diff = $_SERVER['patch']; /* Let's say that someone pasted a patch to html form */

$errors = '';

$new_article = xdiff_string_patch($old_article, $diff, XDIFF_PATCH_NORMAL, $errors);
if (is_string($new_article)) {
    echo "New article:\n";
    echo $new_article;
}

if (strlen($errors)) {
    echo "Rejects: \n";
    echo $errors;
}

?>
```

Returns a patched string.

See also **xdiff_file_patch()**.

CXXXV. Funciones de intérprete XML

Introducción

Acerca de XML

XML (eXtensible Markup Language) es un formato de información para el intercambio de documentos estructurado en la "Web" Es un estándar definido por el consorcio de la "World Wide Web" (W3C). Se puede encontrar información sobre XML y tecnologís relacionadas en http://www.w3.org/XML/.

Instalación

Esta extensión usa expat, que se puede encontrar en http://www.jclark.com/xml/expat.html. El Makefile que viene con expat no crea una biblioteca por defecto, se puede usar esta regla de make para eso:

```
libexpat.a: $(OBJS)
        ar -rc $@ $(OBJS)
        ranlib $@
```

Se puede conseguir un paquete RPM de expat en http://sourceforge.net/projects/expat/.

Nota que si se usa Apache-1.3.7 o posterior, ya tienes la biblioteca requerida expat. Simplemente, configura PHP usando *--with-xml* (sin ninguna ruta adicional) y usará automáticamente la biblioteca expat incluida en Apache.

En UNIX, ejecuta **configure** con la opción *--with-xml*. La biblioteca expat debería ser instalada en algún lugar donde el compilador pueda encontrarlo. Si se compila PHP como un módulo para Apache 1.3.9 o posterior, PHP automáticamente usará la biblioteca integrada expat de Apache. Puede necesitar establecer *CPPFLAGS* y *LDFLAGS* en su entorno antes de ejecutar "configure" si se ha instalado expat en algún lugar exótico.

Compila PHP. *¡Ta-tam!* Ya debería estar.

Sobre Esta Extensión

Esta extensión de PHP implementa soporte para expat de James Clarkin en PHP. Este conjunto de herramientas permite interpretar, pero no validar, documentos XML. Soporta tres codificaciones de caracteres fuente, también proporcionados por PHP: *US-ASCII*, *ISO-8859-1* y *UTF-8*. *UTF-16* no está soportado.

Esta extensión permite crear intérpretes de XML y definir entonces *gestores* para diferentes eventos XML. Cada intérprete XML tiene también unos cuantos parámetros que se pueden ajustar.

Los gestores de eventos XML definidos son:

Tabla 1. Gestores de XML soportados

Función PHP para establecer gestor	Descripción del evento
xml_set_element_handler()	Los eventos de elemento ("element") se producen cuando el intérprete XML encuentra etiquetas de comienzo o fin. Hay gestores separados para etiquetas de comienzo y etiquetas de fin.
xml_set_character_data_handler()	La información de caracteres es, por definición, todo el contenido no "marcado" de los documentos XML, incluidos los espacios en blanco entre etiquetas. Nota que el intérprete XML no añade o elimina ningún espacio en blanco, depende de la aplicación (de ti) decidir si el espacio en blanco es significativo.

Función PHP para establecer gestor	Descripción del evento
xml_set_processing_instruction_handler()	Los programadores de PHP deberían estar ya familiarizados con las instrucciones de procesado (PI). <?php ?> es una instrucción de procesado, donde *php* se denomina el "objetivo de procesado". El manejo de éstos es específico a cada aplicación, salvo que todos los objetivos PI que comienzan con "XML" están reservados.
xml_set_default_handler()	Todo lo que no va a otro gestor, va al gestor por defecto. Se tendrán en el gestor por defecto cosas como las declaraciones de tipos de documento y XML.
xml_set_unparsed_entity_decl_handler()	Este gestor se llamará para la declaración de una entidad no analizada (NDATA).
xml_set_notation_decl_handler()	Este gestor se llama para la declaración de una anotación.
xml_set_external_entity_ref_handler()	Este gestor se llama cuando el intérprete XML encuentra una referencia a una entidad general interpretada externa. Puede ser una referencia a un archivo o URL, por ejemplo. Ver el ejemplo de entidad externa para demostración.

Case Folding

Las funciones manejadoras de elementos pueden tomar sus nombres de elementos "*case-folded*". Case-folding se define en el estándar XML como "un proceso aplicado a una secuencia de caracteres, en el cual aquellos identificados como sin-mayúsculas son reemplazados por sus ñalentes en mayúsculas". En otras palabras, cuando se trata de XML, case-folding simplemente significa poner en mayúsculas.

Por defecto, todos los nombres de elementos que se pasan a las funciones gestoras estan "pasados a mayúsculas". Esta conducta puede ser observada y controlada por el analizador XML con las funciones **xml_parser_get_option()** y **xml_parser_set_option()**, respectivamente.

Códigos de Error

Las siguientes constantes se definen para códigos de error XML (como los devuelve **xml_parse()**):

XML_ERROR_NONE
XML_ERROR_NO_MEMORY
XML_ERROR_SYNTAX
XML_ERROR_NO_ELEMENTS
XML_ERROR_INVALID_TOKEN
XML_ERROR_UNCLOSED_TOKEN
XML_ERROR_PARTIAL_CHAR
XML_ERROR_TAG_MISMATCH
XML_ERROR_DUPLICATE_ATTRIBUTE
XML_ERROR_JUNK_AFTER_DOC_ELEMENT
XML_ERROR_PARAM_ENTITY_REF

XML_ERROR_UNDEFINED_ENTITY
XML_ERROR_RECURSIVE_ENTITY_REF
XML_ERROR_ASYNC_ENTITY
XML_ERROR_BAD_CHAR_REF
XML_ERROR_BINARY_ENTITY_REF
XML_ERROR_ATTRIBUTE_EXTERNAL_ENTITY_RE
F
XML_ERROR_MISPLACED_XML_PI
XML_ERROR_UNKNOWN_ENCODING
XML_ERROR_INCORRECT_ENCODING
XML_ERROR_UNCLOSED_CDATA_SECTION
XML_ERROR_EXTERNAL_ENTITY_HANDLING

Codificación de caracteres

La extension XML de PHP soporta el conjunto de caracteres Unicode a través de diferentes *codificaciones de caracteres*. Hay dos tipos de codificaciones de caracteres, *coficación de fuente* y *codificación de destino*. La representación interna de PHP del documento está siempre codificada con *UTF-8*.

La codificación de fuente se hace cuando un documento XML es interpretado. Al crear un intérprete XML , se puede especificar una codificación de fuente (esta codificación no se puede cambiar má tarde durante el tiempo de vida del intérprete XML). Las codificaciones de fuente soportadas son *ISO-8859-1*, *US-ASCII* y *UTF-8*. Las dos primeras son codificaciones de byte-único, lo que significa que cada carácter se representa por un solo byte. *UTF-8* puede codificar caracteres compuestos por un número variable de bits (hasta 21) en de uno a cuatro bytes. La codificación fuente por defecto usada por PHP es *ISO-8859-1*.

La codificación de destino se hace cuando PHP pasa datos a las funciones gestoras XML. Cuando se crea un intérprete XML, la codificación de destino se crea igual a la codificación de fuente, pero se puede cambiar en cualquier momento. La codificación de destino afectará a la información de los caracteres así como a los nombres de las etiquetas y a los objetivos de instrucciones de procesado.

Si el intérprete XML encuentra caracteres fuera del rango que su codificación de fuente es capaz de representar, devolverá un error.

Si PHP encuentra caracteres en el documento XML interpretado que no pueden ser representados en la codificación de destino elegida, los caracteres problema serán "degradados". Actualmente, esto significa que tales caracteres se reemplazan por un signo de interrogación.

Algunos Ejemplos

Aquí hay algunos ejemplos de archivos de comandos PHP que interpretan documentos XML.

Ejemplos de Estructuras de Elementos XML

Este primer ejemplo muestra la estructura del elemento inicio en un documento con indentación.
Ejemplo 1. Muestra la Estructura del Elemento XML

```
$file = "data.xml";
$depth = array();

function startElement($parser, $name, $attrs) {
    global $depth;
    for ($i = 0; $i < $depth[$parser]; $i++) {
        print "  ";
    }
    print "$name\n";
    $depth[$parser]++;
}

function endElement($parser, $name) {
    global $depth;
    $depth[$parser]--;
}

$xml_parser = xml_parser_create();
xml_set_element_handler($xml_parser, "startElement", "endElement");
if (!($fp = fopen($file, "r"))) {
    die("could not open XML input");
}

while ($data = fread($fp, 4096)) {
    if (!xml_parse($xml_parser, $data, feof($fp))) {
        die(sprintf("XML error: %s at line %d",
                    xml_error_string(xml_get_error_code($xml_parser)),
                    xml_get_current_line_number($xml_parser)));
    }
}
xml_parser_free($xml_parser);
```

Ejemplo de Mapeo de Etiquetas XML

Ejemplo 2. Traduciendo XML a HTML

Este ejemplo transforma etiquetas de un documento XML directamente a etiquetas HTML. Los elementos no encontrados en el "array de traducción ("map array") son ignorados. Por supuesto, este ejemplo solamente funcionará con un tipo de documentos XML específico.

```
$file = "data.xml";
$map_array = array(
    "BOLD"     => "B",
    "EMPHASIS" => "I",
    "LITERAL"  => "TT"
);

function startElement($parser, $name, $attrs) {
    global $map_array;
    if ($htmltag = $map_array[$name]) {
        print "<$htmltag>";
    }
}

function endElement($parser, $name) {
    global $map_array;
    if ($htmltag = $map_array[$name]) {
        print "</$htmltag>";
    }
}

function characterData($parser, $data) {
    print $data;
}

$xml_parser = xml_parser_create();
// usa case-folding para que estemos seguros de encontrar la etiqueta
// en $map_array
xml_parser_set_option($xml_parser, XML_OPTION_CASE_FOLDING, true);
xml_set_element_handler($xml_parser, "startElement", "endElement");
xml_set_character_data_handler($xml_parser, "characterData");
if (!($fp = fopen($file, "r"))) {
    die("could not open XML input");
}

while ($data = fread($fp, 4096)) {
    if (!xml_parse($xml_parser, $data, feof($fp))) {
        die(sprintf("XML error: %s at line %d",
                    xml_error_string(xml_get_error_code($xml_parser)),
                    xml_get_current_line_number($xml_parser)));
    }
}
xml_parser_free($xml_parser);
```

Ejemplo de Entidad Externa XML

Este ejemplo resalta el código XML. Ilustra cómo usar un gestor de referencia de entidades extenas para incluir y analizar otros documentos, así como cuúntos PIs pueden ser procesados, y un modo de determinar "confianza" para PIs que contienen código.

Los documentos XML que se pueden usar en este ejemplo se encuentran bajo el ejemplo (`xmltest.xml` y `xmltest2.xml`.)

Ejemplo 3. Ejemplo de Entidades Externas

```
$file = "xmltest.xml";

function trustedFile($file) {
    // solamente confía en archivos locales que nos pertenezcan
    if (!eregi("^([a-z]+)://", $file)
        && fileowner($file) == getmyuid()) {
            return true;
    }
    return false;
}

function startElement($parser, $name, $attribs) {
    print "&lt;<font color=\"#0000cc\">$name</font>";
    if (sizeof($attribs)) {
        while (list($k, $v) = each($attribs)) {
            print " <font color=\"#009900\">$k</font>=\"<font
                   color=\"#990000\">$v</font>\"";
        }
    }
    print "&gt;";
}

function endElement($parser, $name) {
    print "&lt;/<font color=\"#0000cc\">$name</font>&gt;";
}

function characterData($parser, $data) {
    print "<b>$data</b>";
}

function PIHandler($parser, $target, $data) {
    switch (strtolower($target)) {
        case "php":
            global $parser_file;
            // Si el documento analizado es "de confianza", diremos
            // que es seguro ejecutar código PHP en su interior.
            // Si no, en vez de ello mostrará el código.
            if (trustedFile($parser_file[$parser])) {
                eval($data);
            } else {
                printf("Untrusted PHP code: <i>%s</i>",
                        htmlspecialchars($data));
            }
            break;
    }
}

function defaultHandler($parser, $data) {
    if (substr($data, 0, 1) == "&" && substr($data, -1, 1) == ";") {
        printf('<font color="#aa00aa">%s</font>',
                htmlspecialchars($data));
    } else {
        printf('<font size="-1">%s</font>',
                htmlspecialchars($data));
    }
}

function externalEntityRefHandler($parser, $openEntityNames, $base, $systemId,
                                   $publicId) {
    if ($systemId) {
        if (!list($parser, $fp) = new_xml_parser($systemId)) {
            printf("Could not open entity %s at %s\n", $openEntityNames,
                   $systemId);
            return false;
        }
        while ($data = fread($fp, 4096)) {
            if (!xml_parse($parser, $data, feof($fp))) {
                printf("XML error: %s at line %d while parsing entity %s\n",
                       xml_error_string(xml_get_error_code($parser)),
                       xml_get_current_line_number($parser), $openEntityNames);
                xml_parser_free($parser);
                return false;
```

Ejemplo 4. xmltest.xml

```
<?xml version='1.0'?>
<!DOCTYPE chapter SYSTEM "/just/a/test.dtd" [
<!ENTITY plainEntity "FOO entity">
<!ENTITY systemEntity SYSTEM "xmltest2.xml">
]>
<chapter>
 <TITLE>Title &plainEntity;</TITLE>
 <para>
  <informaltable>
   <tgroup cols="3">
    <tbody>
     <row><entry>a1</entry><entry morerows="1">b1</entry><entry>c1</entry></row>
     <row><entry>a2</entry><entry>c2</entry></row>
     <row><entry>a3</entry><entry>b3</entry><entry>c3</entry></row>
    </tbody>
   </tgroup>
  </informaltable>
 </para>
 &systemEntity;
 <sect1 id="about">
  <title>About this Document</title>
  <para>
   <!-- this is a comment -->
   <?php print 'Hi!  This is PHP version '.phpversion(); ?>
  </para>
 </sect1>
</chapter>
```

Este archivo se incluye desde `xmltest.xml`:

Ejemplo 5. xmltest2.xml

```
<?xml version="1.0"?>
<!DOCTYPE foo [
<!ENTITY testEnt "test entity">
]>
<foo>
   <element attrib="value"/>
   &testEnt;
   <?php print "This is some more PHP code being executed."; ?>
</foo>
```

Tabla de contenidos

xml_set_object -- Usa un analizador XML dentro de un objecto
xml_set_processing_instruction_handler -- Establece el gestor de instrucciones de procesado (PI)
xml_set_start_namespace_decl_handler -- Set up start namespace declaration handler
xml_set_unparsed_entity_decl_handler -- Establece un gestor de declaraciones de entidades no analizadas

utf8_decode

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

utf8_decode -- Convierte una cadena codificada UTF-8 a ISO-8859-1

Descripción

string **utf8_decode** (string data)

Esta función decodifica*data*, asume codificación *UTF-8* , a *ISO-8859-1*.

Mira **utf8_encode()** para una explicación de codificación UTF-8.

utf8_encode

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

utf8_encode -- codifica una cadena ISO-8859-1 a UTF-8

Descripción

string **utf8_encode** (string data)

Esta función codifica la cadena *data* a *UTF-8*, y devuelve la versión codificada. *UTF-8* es un mecanismo estándar usado por Unicodepara codificar valores de *caracteres amplios* en un chorro de bytes. *UTF-8* es transparente a caracteres de ASCII plano, es auto-sincronizado (significa que es posible para un programa averiguar dónde comienzan los caracteres en el chorro de bytes) y se puede usar con funciones de comparación de cadenas normales para ordenar y otros fines. PHP codifica caracteres *UTF-8* en hasta cuatro bytes, como esto:

Tabla 1. Codificación UTF-8

bytes	bits	representación
1	7	0bbbbbbb
2	11	110bbbbb 10bbbbbb
3	16	1110bbbb 10bbbbbb 10bbbbbb
4	21	11110bbb 10bbbbbb 10bbbbbb 10bbbbbb

Cada *b* representa un bit que puede ser usado para almacenar datos de caracteres.

xml_error_string

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

xml_error_string -- obtiene la cadena de error del analizador XML

Descripción

string **xml_error_string** (int code)

code

Un código de error de **xml_get_error_code()**.

Devuelve una cadena con una descripción textual del código de error *code*, o **FALSE** si no se encontró descripción.

xml_get_current_byte_index

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

xml_get_current_byte_index -- obtiene el índice del byte actual para un analizador XML

Descripción

int **xml_get_current_byte_index** (int parser)

parser

Una referencia al analizador XML del que obtener el índice del byte.

Esta función devuelve **FALSE** si *parser* no referencia un analizador válido, o si no devuelve en qué índice de byte se encuentra el buffer de datos del analizador (empezando en 0).

xml_get_current_column_number

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

xml_get_current_column_number -- Obtiene el número de columna actual para un analizador XML.

Descripción

int **xml_get_current_column_number** (int parser)

parser

Una referencia al analizador XML del que obtener el número de columna.

Esta función devuelve **FALSE** si *parser* no referencia un analizador válido, o si no devuelve en qué columna de la línea actual (como se obtiene de **xml_get_current_line_number()**) en la que se encuentra el analizador.

xml_get_current_line_number

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

xml_get_current_line_number -- obtiene el número de línea actual de un analizador XML

Descripción

int **xml_get_current_line_number** (int parser)

parser

Una referencia al analizador XML del que obtener el número de línea.

Esta función devuelve **FALSE** si *parser* no referencia un analizador válido, o si no devuelve en qué línea se encuentra actualmente el buffer de datos del analizador.

xml_get_error_code

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

xml_get_error_code -- obtiene el código de error del analizador XML

Descripción

int **xml_get_error_code** (int parser)

parser

Una referencia al analizador XML del que obtener el código de error.

Esta función devuelve **FALSE** si *parser* no referencia un analizador válido, o si no devuelve uno de los códigos de error listados en la sección de códigos de error.

xml_parse_into_struct

(PHP 3>= 3.0.8, PHP 4 , PHP 5)

xml_parse_into_struct -- Parse XML data into an array structure

Description

int **xml_parse_into_struct** (resource parser, string data, array &values [, array &index])

This function parses an XML file into 2 parallel array structures, one (*index*) containing pointers to

the location of the appropriate values in the *values* array. These last two parameters must be passed by reference.

Below is an example that illustrates the internal structure of the arrays being generated by the function. We use a simple *note* tag embedded inside a *para* tag, and then we parse this and print out the structures generated:

Ejemplo 1. xml_parse_into_struct() example

```
<?php
$simple = "<para><note>simple note</note></para>";
$p = xml_parser_create();
xml_parse_into_struct($p, $simple, $vals, $index);
xml_parser_free($p);
echo "Index array\n";
print_r($index);
echo "\nVals array\n";
print_r($vals);
?>
```

When we run that code, the output will be:

```
Index array
Array
(
    [PARA] => Array
        (
            [0] => 0
            [1] => 2
        )

    [NOTE] => Array
        (
            [0] => 1
        )

)

Vals array
Array
(
    [0] => Array
        (
            [tag] => PARA
            [type] => open
            [level] => 1
        )

    [1] => Array
        (
            [tag] => NOTE
            [type] => complete
            [level] => 2
            [value] => simple note
        )

    [2] => Array
        (
            [tag] => PARA
            [type] => close
            [level] => 1
        )

)
```

Event-driven parsing (based on the expat library) can get complicated when you have an XML document that is complex. This function does not produce a DOM style object, but it generates structures amenable of being transversed in a tree fashion. Thus, we can create objects representing the data in the XML file easily. Let's consider the following XML file representing a small database of aminoacids information:

Ejemplo 2. moldb.xml - small database of molecular information

```
<?xml version="1.0"?>
<moldb>

    <molecule>
        <name>Alanine</name>
        <symbol>ala</symbol>
        <code>A</code>
        <type>hydrophobic</type>
    </molecule>

    <molecule>
        <name>Lysine</name>
        <symbol>lys</symbol>
        <code>K</code>
        <type>charged</type>
    </molecule>

</moldb>
```

And some code to parse the document and generate the appropriate objects:

Ejemplo 3. parsemoldb.php - parses moldb.xml into an array of molecular objects

```
<?php

class AminoAcid {
    var $name;  // aa name
    var $symbol;    // three letter symbol
    var $code;  // one letter code
    var $type;  // hydrophobic, charged or neutral

    function AminoAcid ($aa)
    {
        foreach ($aa as $k=>$v)
            $this->$k = $aa[$k];
    }
}

function readDatabase($filename)
{
    // read the XML database of aminoacids
    $data = implode("", file($filename));
    $parser = xml_parser_create();
    xml_parser_set_option($parser, XML_OPTION_CASE_FOLDING, 0);
    xml_parser_set_option($parser, XML_OPTION_SKIP_WHITE, 1);
    xml_parse_into_struct($parser, $data, $values, $tags);
    xml_parser_free($parser);

    // loop through the structures
    foreach ($tags as $key=>$val) {
        if ($key == "molecule") {
            $molranges = $val;
            // each contiguous pair of array entries are the
            // lower and upper range for each molecule definition
            for ($i=0; $i < count($molranges); $i+=2) {
                    $offset = $molranges[$i] + 1;
                $len = $molranges[$i + 1] - $offset;
                $tdb[] = parseMol(array_slice($values, $offset, $len));
            }
        } else {
            continue;
        }
    }
    return $tdb;
}

function parseMol($mvalues)
{
    for ($i=0; $i < count($mvalues); $i++) {
        $mol[$mvalues[$i]["tag"]] = $mvalues[$i]["value"];
    }
    return new AminoAcid($mol);
}

$db = readDatabase("moldb.xml");
echo "** Database of AminoAcid objects:\n";
print_r($db);

?>
```

After executing `parsemoldb.php`, the variable *$db* contains an array of **AminoAcid** objects, and the output of the script confirms that:

```
** Database of AminoAcid objects:
Array
(
    [0] => aminoacid Object
        (
            [name] => Alanine
            [symbol] => ala
            [code] => A
            [type] => hydrophobic
        )

    [1] => aminoacid Object
        (
            [name] => Lysine
            [symbol] => lys
            [code] => K
            [type] => charged
        )

)
```

xml_parse

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

xml_parse -- comienza a analizar un documento XML

Descripción

int **xml_parse** (int parser, string data [, int isFinal])

parser

Una referencia al analizador XML que se va a utilizar.

data

Conjunto de información que se analizará. Un documento puede ser analizado por trozos llamando varias veces a **xml_parse()** con nueva información, siempre que se establezca el parámetro *isFinal* y sea **TRUE** cuando el último dato sea analizado.

isFinal (optional)

Si existe y es **TRUE**, *data* es el último pedazo de información envíado en este análisis.

Cuando el documento XML es analizado, se hacen llamadas a los gestores para los eventos configurados tantas veces como sea necesario, después de que esta función devuelva **TRUE** o **FALSE**.

Devuelve **TRUE** si el análisis se realiza con éxito, **FALSE** si no tiene éxito, o si *parser* no referencia a un analizador válido. Para análisis fallidos, se puede recuperar información de error con **xml_get_error_code()**, **xml_error_string()**, **xml_get_current_line_number()**, **xml_get_current_column_number()** y **xml_get_current_byte_index()**.

xml_parser_create_ns

(PHP 4 >= 4.0.5, PHP 5)

xml_parser_create_ns -- Create an XML parser with namespace support

Description

resource **xml_parser_create_ns** ([string encoding [, string separator]])

xml_parser_create_ns() creates a new XML parser with XML namespace support and returns a resource handle referencing it to be used by the other XML functions.

With a namespace aware parser tag parameters passed to the various handler functions will consist of namespace and tag name separated by the string specified in *seperator* or *':'* by default.

The optional *encoding* specifies the character encoding for the input/output in PHP 4. Starting from PHP 5, the input encoding is automatically detected, so that the *encoding* parameter specifies only the output encoding. In PHP 4, the default output encoding is the same as the input charset. In PHP 5.0.0 and 5.0.1, the default output charset is ISO-8859-1, while in PHP 5.0.2 and upper is UTF-8. The supported encodings are *ISO-8859-1*, *UTF-8* and *US-ASCII*.

See also **xml_parser_create()**, and **xml_parser_free()**.

xml_parser_create

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

xml_parser_create -- crea un analizador de XML

Descripción

int **xml_parser_create** ([string encoding])

encoding (opcional)

> Qué codificación de caracteres debería usar el analizador. Las siguientes codificación de caracteres están soportadas:
>
> *ISO-8859-1* (por defecto)
> *US-ASCII*
> *UTF-8*

Esta función crea un analizador XML y devuelve un índice para usarlo con otras funciones XML. Devuelve **FALSE** en caso de fallo.

xml_parser_free

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

xml_parser_free -- Libera un analizador XML

Descripción

string **xml_parser_free** (int parser)

parser

Una referencia al analizador XML que se liberará.

Esta función devuelve **`FALSE`** si *parser* no referencia un analizador válido, o si no libera el analizador y devuelve **`TRUE`**.

xml_parser_get_option

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

xml_parser_get_option -- obtiene las opciones de un analizador XML

Descripción

mixed **xml_parser_get_option** (int parser, int option)

parser

Una referencia al analizador XML del que obtener opciones.

option

Qué opción recuperar. Ver **xml_parser_set_option()** para una lista de opciones.

Esta función devuelve **`FALSE`** si *parser* no referencia un analizador válido, o si la opción no pudo ser establecida. Si no, se devuelve la opción.

Mirar **xml_parser_set_option()** para la lista de opciones.

xml_parser_set_option

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

xml_parser_set_option -- establece las opciones de un analizador XML

Descripción

int **xml_parser_set_option** (int parser, int option, mixed value)

parser

Una referencia al analizador XML en el que establecer opciones.

option

Opción que se establecerá. Ver más abajo.

value

El nuevo valor de la opción.

Esta función devuelve **FALSE** si *parser* no referencia un analizador válido, o si la opción no pudo ser establecida. Si no, la opción se establece y devuelve **TRUE**.

Las opciones siguientes están disponibles:

Tabla 1. Opciones de analizador XML

Opción constante	Tipo de Datos	Descripción
XML_OPTION_CASE_FOLDING	integer	Controla si case-folding se habilita para este analizador XML. Habilitado por defecto.
XML_OPTION_TARGET_ENCODING	string	Establece qué codificación destino se usa en este analizador XML. Por defecto, esta puesta al mismo que la codificación fuente usada por **xml_parser_create()**. Las codificaciones de destino soportadas son *ISO-8859-1*, *US-ASCII* y *UTF-8*.

xml_set_character_data_handler

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

xml_set_character_data_handler -- Establece gestores de datos de caracteres

Descripción

int **xml_set_character_data_handler** (int parser, string handler)

Establece la función gestora de datos de caracteres para el analizador XML *parser*. *handler* es un string que contiene el nombre de la función que debe existir cuando **xml_parse()** es llamado por *parser*.

La función nombrada en *handler* debe aceptar dos parámetros: ***handler*** (int parser, string data)

parser

El primer parámetro, *parser*, es una referencia al analizador XML que llama al gestor.

data

El segundo parámetro, *data*, contiene los datos caracteres como string.

Si una función gestora se establece como la cadena vacía, o **FALSE**, el gestor en cuestión se deshabilita.

Se devuelve **TRUE** si se estableció el gestor, **FALSE** si *parser* no es un analizador.

En la actualidad no hay soporte para gestores objeto/método.

xml_set_default_handler

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

xml_set_default_handler -- set up default handler

Descripción

int **xml_set_default_handler** (int parser, string handler)

Establece la función gestora por defecto para un analizador XML *parser*. *handler* es un string que contiene el nombre de la función que debe existir cuando **xml_parse()** es llamado por *parser*.

La función nombrada en *handler* debe aceptar dos parámetros: ***handler*** (int parser, string data)

parser

El primer parámetro, *parser*, es una referencia al analizador XML que llama al gestor.

data

El segundo parámetro, *data*, contiene los caracteres de dato. Esto puede ser la declaración XML, la declaración de tipo de documento, entidades u otros datos para los cuales no existe otro gestor.

Si una función gestora se establece como la cadena vacía, o **FALSE**, el gestor en cuestión se deshabilita.

Se devuelve **TRUE** si se estableció el gestor, **FALSE** si *parser* no es un analizador.

En la actualidad no hay soporte para gestores objeto/método.

xml_set_element_handler

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

xml_set_element_handler -- establece gestores de los elementos principio y fin

Descripción

int **xml_set_element_handler** (int parser, string startElementHandler, string endElementHandler)

Establece las funciones de gestion de elementos para el analizador XML *parser*. *startElementHandler* y *endElementHandler* son strings que contienen los nombres de las funciones que deben existir cuando **xml_parse()** es llamado por *parser*.

La función denominada *startElementHandler* debe aceptar tres parámetros: ***startElementHandler***

(int parser, string name, string attribs)

parser

El primer parámetro, *parser*, es una referencia al analizador XML que llama al gestor.

name

El segundo parámetro, *name*, contiene el nombre del elemento para el que se llama a este gestor. Si la propiedad de case-folding tiene efecto para este analizador, el nombre del elemento estará en mayúsculas.

attribs

El tercer parámetro, *attribs*, contiene un array asociativo con los atributos de los elementos (si hay). Las claves de este array son los nombres de los atributos, los valores son los valores de los atributos. Los nombres de los atributos están en mayúsculas (case-folded) con el mismo criterio que los nombres de los elementos. Los valores de los atributos *no* sufren las consecuencias de case-folding.

El orden original de los atributos se puede recuperar recorriendo *attribs* del modo usual, usando **each()**. La primera clave del array es el el primer atributo, y así sucesivamente.

La función llamada *endElementHandler* debe aceptar dos parámetros: ***endElementHandler*** (int parser, string name)

parser

El primer parámetro, *parser*, es una referencia al analizador XML que llama al gestor.

name

El segundo parámetro, *name*, contiene el nombre del elemento para el que se llama a este gestor. Si la propiedad de case-folding tiene efecto para este analizador, el nombre del elemento estará en mayúsculas.

Si una función gestora se establece como la cadena vacía, o **FALSE**, el gestor en cuestión se deshabilita.

Se devuelve **TRUE** si se establecieron los gestores, **FALSE** si *parser* no es un analizador.

En la actualidad no hay soporte para gestores objeto/método.

xml_set_end_namespace_decl_handler

(PHP 4 >= 4.0.5, PHP 5)

xml_set_end_namespace_decl_handler -- Set up end namespace declaration handler

Description

bool **xml_set_end_namespace_decl_handler** (resource parser, callback handler)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

Nota: En lugar de un nombre de función, se puede proporcionar una matriz que contenga una referencia a un objeto o el nombre de un método.

xml_set_external_entity_ref_handler

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

xml_set_external_entity_ref_handler -- Establece gestores de referencia de entidades externas

Descripción

int **xml_set_external_entity_ref_handler** (int parser, string handler)

Establece la función gestora de declaraciones de notación para el analizador XML *parser*. *handler* es un string que contiene el nombre de una función que debe existir cuando **xml_parse()** es llamado por *parser*.

La función llamada por *handler* debe aceptar cinco parámetros, y debería devolver un valor entero. Si el valor devuelto desde el gestor (handler) es falso (lo cual ocurrirá si no se devuelve un valor), el analizador XML dejará de analizar y **xml_get_error_code()** devolverá XML_ERROR_EXTERNAL_ENTITY_HANDLING. int ***handler*** (int parser, string openEntityNames, string base, string systemId, string publicId)

parser

El primer parámetro, *parser*, es una referencia al analizador XML que llama al gestor.

openEntityNames

El segundo parámetro, *openEntityNames*, es una lista, separada por espacios, de los nombres de las entidades que se abren para el análisis de esta entidad (incluido el nombre de la entidad referenciada).

base

Esta es la base para resolver el identificador de sistema (*systemid*) de la entidad externa. En la actualidad este parámetro es siempre la cadena vacía.

systemId

El cuarto parámetro, *systemId*, es el identificador del sistema tal como se especificó en la declaración de la entidad.

publicId

El quinto parámetro, *publicId*, es el identificador público como se especificó en la declaración de la entidad, o una cadena vacía si no se especificó ninguno; el espacio en blanco en el identificador público se habrá normalizado como se requiere en las especificaciones XML.

Si una función gestora se establece como la cadena vacía, o **FALSE**, el gestor en cuestión se deshabilita.

Se devuelve **TRUE** si se estableció el gestor, **FALSE** si *parser* no es un analizador.

En la actualidad no hay soporte para gestores objeto/método.

xml_set_notation_decl_handler

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

xml_set_notation_decl_handler -- Establece gestores de declaraciones de notación

Descripción

int **xml_set_notation_decl_handler** (int parser, string handler)

Establece las funciones gestoras de declaraciones de notación para el analizador XML *parser*. *handler* es un string que contiene el nombre de una función que debe existir cuando **xml_parse()** es llamado por *parser*.

Una declaración de notación es parte del DTD del documento y tiene el siguiente formato:

```
<!NOTATION name
     {systemId | publicId}
     >
```

Ver la sección 4.7 de las especificaciones XML 1.0 para la definición de declaraciones de notación.

La función llamada por *handler* debe aceptar cinco parámetros: ***handler*** (int parser, string notationName, string base, string systemId, string publicId)

parser

El primer parámetro, *parser*, es una referencia al analizador XML que llama al gestor.

notationName

Este es el *nombre* de la notación, como se describió arriba en el formato de notación.

base

Esta es la base para resolver el identificador de sistema (*systemId*) de la declaración. En la actualidad este parámetro es siempre la cadena vacía.

systemId

Identificador de sistema de la declaración de notación externa.

publicId

Identificador público de la declaración de notación externa.

Si una función gestora se establece como la cadena vacía, o **FALSE**, el gestor en cuestión se deshabilita.

Se devuelve **TRUE** si se estableció el gestor, **FALSE** si *parser* no es un analizador.

En la actualidad no hay soporte para gestores objeto/método.

xml_set_object

(PHP 4 , PHP 5)

xml_set_object -- Usa un analizador XML dentro de un objecto

Descripción

void **xml_set_object** (int parser, object &object)

Esta función hace a *parser* utilizable dentro de *object*. Todas las funciones de callback establecidas por **xml_set_element_handler()** etc se asumen como métodos de *object*.

```
<?php

class xml  {
var $parser;

function xml() {
    $this->parser = xml_parser_create();
    xml_set_object($this->parser,$this);
    xml_set_element_handler($this->parser,"tag_open","tag_close");
    xml_set_character_data_handler($this->parser,"cdata");
}

function parse($data) {
    xml_parse($this->parser,$data);
}

function tag_open($parser,$tag,$attributes) {
    var_dump($parser,$tag,$attributes);
}

function cdata($parser,$cdata) {
    var_dump($parser,$cdata);
}

function tag_close($parser,$tag) {
    var_dump($parser,$tag);
}

} // end of class xml

$xml_parser = new xml();
$xml_parser->parse("&lt;A ID=\"hallo\">PHP&lt;/A>");
?>
```

Nota: xml_set_object() es gestionable a partir de PHP 4.0.

xml_set_processing_instruction_handler

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

xml_set_processing_instruction_handler -- Establece el gestor de instrucciones de procesado (PI)

Descripción

int **xml_set_processing_instruction_handler** (int parser, string handler)

Establece ela función de gestión de instrucciones de procesado (PI) para el analizador XML *parser*. *handler* es un string que contiene el nombre de una función que debe existir cuando **xml_parse()** es llamada por *parser*.

Una instrucción de procedado tiene el siguiente formato:

```
<?
        target
        data?>
```

Puedes poner código PHP en esa etiqueta, pero ten en cuenta una limitación: en una PI XML, la etiqueta de fin de la PI (*?*>) no puede ser citada, por lo que esta secuencia de caracteres no debería aparecer en el código PHP que insertes con las PIs en documentos XML. Si lo hace, el resto del código PHP, así como la etiqueta de fin de PI "real", serán tratados como datos de caracteres.

La función nombrada en *handler* debe aceptar tres parámetros: ***handler*** (int parser, string target, string data)

parser

El primer parámetro, *parser*, es una referencia al analizador XML que llama al gestor.

target

El segundo parámetro, *target*, contiene el objetivo PI.

data

El tercer parámetro, *data*, contiene los datos PI.

Si una función gestora se establece como la cadena vacía, o **FALSE**, el gestor en cuestión se deshabilita.

Se devuelve **TRUE** si se estableció el gestor, **FALSE** si *parser* no es un analizador.

En la actualidad no hay soporte para gestores objeto/método.

xml_set_start_namespace_decl_handler

(PHP 4 >= 4.0.5, PHP 5)

xml_set_start_namespace_decl_handler -- Set up start namespace declaration handler

Description

bool **xml_set_start_namespace_decl_handler** (resource parser, callback handler)

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

Nota: En lugar de un nombre de función, se puede proporcionar una matriz que contenga una referencia a un objeto o el nombre de un método.

xml_set_unparsed_entity_decl_handler

(PHP 3>= 3.0.6, PHP 4 , PHP 5)

xml_set_unparsed_entity_decl_handler -- Establece un gestor de declaraciones de entidades no analizadas

Descripción

int **xml_set_unparsed_entity_decl_handler** (int parser, string handler)

Establece la función gestora de declaración de entidades no analizadas para el analizador XML *parser*. *handler* es una cadena que contiene el nombre de una función que debe existir cuando **xml_parse()** es llamada por *parser*.

Este gestor será llamado si el analizador XML encuentra una declaración de entidades externas con una declaración NDATA, como la siguiente:

```
<!ENTITY name {publicId | systemId}
        NDATA notationName>
```

Mira la sección 4.2.2 de las especificaciones XML 1.0 para la definición de entidades externas de notación declarada.

La función nombrada en *handler* debe aceptar seis parámetros: ***handler*** (int parser, string entityName, string base, string systemId, string publicId, string notationName)

parser

El primer parámetro, *parser*, es una referencia al analizador XML que llama al gestor.

entityName

El nombre de la entidad que va a ser definida.

base

Esta es la base para resolver el identificador de sistema (*systemId*) de la entidad externa. Actualmente este parámetro siempre será una cadena vacía.

systemId

Identificador de Sistema para la entidad externa.

publicId

Identificador público para la entidad externa.

notationName

Nombre de la notación de esta entidad (ver **xml_set_notation_decl_handler()**).

Si una función gestora se establece como la cadena vacía, o **FALSE**, el gestor en cuestión se deshabilita.

Se devuelve **TRUE** si se estableció el gestor, **FALSE** si *parser* no es un analizador.

En la actualidad no hay soporte para gestores objeto/método.

CXXXVI. XML-RPC Functions

Introducción

These functions can be used to write XML-RPC servers and clients. You can find more information about XML-RPC at http://www.xmlrpc.com/, and more documentation on this extension and its functions at http://xmlrpc-epi.sourceforge.net/.

Aviso
Esta extensión es *EXPERIMENTAL*. Esto significa que el comportamiento de esta extensión, los nombre de sus funciones y en definitiva TODO lo documentado sobre esta extensión, puede cambiar en una futura versión de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Requirimientos

No se necesitan bibliotecas externas para construir esta extensión

Instalación

XML-RPC support in PHP is not enabled by default. You will need to use the *--with-xmlrpc[=DIR]* configuration option when compiling PHP to enable XML-RPC support. This extension is bundled into PHP as of 4.1.0.

Configuración en tiempo de ejecución

El comportamiento de estas funciones está afectado por los valores definidos en `php.ini`.

Tabla 1. XML-RPC configuration options

Name	Default	Changeable
xmlrpc_errors	"0"	PHP_INI_SYSTEM
xmlrpc_error_number	"0"	PHP_INI_ALL

For further details and definitions of the PHP_INI_* constants, see the Apéndice H.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Esta extensión no tiene ninguna constante definida.

Tabla de contenidos

xmlrpc_decode_request

(PHP 4 >= 4.1.0, PHP 5)

xmlrpc_decode_request -- Decodes XML into native PHP types

Description

array **xmlrpc_decode_request** (string xml, string &method [, string encoding])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

xmlrpc_decode

(PHP 4 >= 4.1.0, PHP 5)

xmlrpc_decode -- Decodes XML into native PHP types

Description

array **xmlrpc_decode** (string xml [, string encoding])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.
Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

xmlrpc_encode_request

(PHP 4 >= 4.1.0, PHP 5)

xmlrpc_encode_request -- Generates XML for a method request

Description

string **xmlrpc_encode_request** (string method, mixed params [, array output_options])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.
Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

xmlrpc_encode

(PHP 4 >= 4.1.0, PHP 5)

xmlrpc_encode -- Generates XML for a PHP value

Description

string **xmlrpc_encode** (mixed value)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

xmlrpc_get_type

(PHP 4 >= 4.1.0, PHP 5)

xmlrpc_get_type -- Gets xmlrpc type for a PHP value

Description

string **xmlrpc_get_type** (mixed value)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

This function is especially useful for base64 and datetime strings.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

xmlrpc_is_fault

(PHP 4 >= 4.3.0, PHP 5)

xmlrpc_is_fault -- Determines if an array value represents an XMLRPC fault

Description

bool **xmlrpc_is_fault** (array arg)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

xmlrpc_parse_method_descriptions

(PHP 4 >= 4.1.0, PHP 5)

xmlrpc_parse_method_descriptions -- Decodes XML into a list of method descriptions

Description

array **xmlrpc_parse_method_descriptions** (string xml)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

xmlrpc_server_add_introspection_data

(PHP 4 >= 4.1.0, PHP 5)

xmlrpc_server_add_introspection_data -- Adds introspection documentation

Description

int **xmlrpc_server_add_introspection_data** (resource server, array desc)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

xmlrpc_server_call_method

(PHP 4 >= 4.1.0, PHP 5)

xmlrpc_server_call_method -- Parses XML requests and call methods

Description

mixed **xmlrpc_server_call_method** (resource server, string xml, mixed user_data [, array output_options])

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

xmlrpc_server_create

(PHP 4 >= 4.1.0, PHP 5)

xmlrpc_server_create -- Creates an xmlrpc server

Description

resource **xmlrpc_server_create** (void)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

xmlrpc_server_destroy

(PHP 4 >= 4.1.0, PHP 5)

xmlrpc_server_destroy -- Destroys server resources

Description

int **xmlrpc_server_destroy** (resource server)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

xmlrpc_server_register_introspection_callback

(PHP 4 >= 4.1.0, PHP 5)

xmlrpc_server_register_introspection_callback -- Register a PHP function to generate documentation

Description

bool **xmlrpc_server_register_introspection_callback** (resource server, string function)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

xmlrpc_server_register_method

(PHP 4 >= 4.1.0, PHP 5)

xmlrpc_server_register_method -- Register a PHP function to resource method matching method_name

Description

bool **xmlrpc_server_register_method** (resource server, string method_name, string function)

Aviso

Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

xmlrpc_set_type

(PHP 4 >= 4.1.0, PHP 5)

xmlrpc_set_type -- Sets xmlrpc type, base64 or datetime, for a PHP string value

Description

bool **xmlrpc_set_type** (string &value, string type)

Aviso
Esta función es *EXPERIMENTAL*. Esto significa que el comportamiento de esta función, el nombre de esta función y en definitiva TODO lo documentado sobre esta función, puede cambiar en una futura version de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

CXXXVII. XSL functions

Introducción

The XSL extension implements the XSL standard, performing XSLT transformations using the libxslt library

---

Requirimientos

This extension uses libxslt which can be found at http://xmlsoft.org/XSLT/. libxslt version 1.0.18 or greater is required.

Instalación

PHP 5 includes the XSL extension by default and can be enabled by adding the argument *--with-xsl [=DIR]* to your configure line. *DIR* is the libxslt installation directory.

Clases predefinidas

XSLTProcessor

Constructor

- XSLTProcessor->__construct() - construct a new XSLTProcessor object

---

Métodos

- XSLTProcessor->getParameter() - Get value of a parameter
- XSLTProcessor->hasExsltSupport() - Determine if PHP has EXSLT support
- XSLTProcessor->importStylesheet() - Import stylesheet
- XSLTProcessor->registerPHPFunctions() - Enables the ability to use PHP functions as XSLT functions
- XSLTProcessor->removeParameter() - Remove parameter
- XSLTProcessor->setParameter() - Set value for a parameter
- XSLTProcessor->transformToDoc() - Transform to DOMDocument
- XSLTProcessor->transformToURI() - Transform to URI
- XSLTProcessor->transformToXML() - Transform to XML

Ejemplos

Many examples in this reference require both an XML and an XSL file. We will use `collection.xml` and `collection.xsl` that contains the following:

Ejemplo 1. collection.xml

```
<collection>
 <cd>
  <title>Fight for your mind</title>
  <artist>Ben Harper</artist>
  <year>1995</year>
 </cd>
 <cd>
  <title>Electric Ladyland</title>
  <artist>Jimi Hendrix</artist>
  <year>1997</year>
 </cd>
</collection>
```

Ejemplo 2. collection.xsl

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:param name="owner" select="'Nicolas Eliaszewicz'"/>
 <xsl:output method="html" encoding="iso-8859-1" indent="no"/>
 <xsl:template match="collection">
  Hey! Welcome to <xsl:value-of select="$owner"/>'s sweet CD collection!
  <xsl:apply-templates/>
 </xsl:template>
 <xsl:template match="cd">
  <h1><xsl:value-of select="title"/></h1>
  <h2>by <xsl:value-of select="artist"/> - <xsl:value-of select="year"/></h2>
  <hr />
 </xsl:template>
</xsl:stylesheet>
```

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

XSL_CLONE_AUTO (integer)

XSL_CLONE_NEVER (integer)

XSL_CLONE_ALWAYS (integer)

Tabla de contenidos

XSLTProcessor->__construct()

XSLTProcessor->__construct() -- Creates a new XSLTProcessor object

Descripción

class **XSLTProcessor** {

__construct ((void);)

}

Creates a new **XSLTProcessor** object.

Ejemplos

Ejemplo 1. Creating an XSLTProcessor

```
<?php

$xsl = new XSLTProcessor();
$xsl->importStyleSheet(DOMDocument::load($xsl_filename));
echo $xsl->transformToXML(DOMDocument::load($xml_filename));

?>
```

XSLTProcessor->getParameter()

XSLTProcessor->getParameter() -- Get value of a parameter

Descripción

class **XSLTProcessor** {

string **getParameter** (string namespaceURI, string localName)

}

Gets a parameter if previously set by XSLTProcessor->setParameter().

Lista de parámetros

namespaceURI

The namespace URI of the XSLT parameter.

localName

The local name of the XSLT parameter.

Valores retornados

The value of the parameter or **`NULL`** if it's not set.

Ver también

XSLTProcessor->setParameter()
XSLTProcessor->removeParameter()

XSLTProcessor->hasExsltSupport()

XSLTProcessor->hasExsltSupport() -- Determine if PHP has EXSLT support

Descripción

class **XSLTProcessor** {

bool **hasExsltSupport** (void)

}

This method determine if PHP was built with the EXSLT library.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ejemplos

Ejemplo 1. Testing EXSLT support

```
<?php

$proc = new XSLTProcessor;
if (!$proc->hasExsltSupport()) {
    die('EXSLT support not available');
}

// do EXSLT stuff here ..

?>
```

XSLTProcessor->importStylesheet()

XSLTProcessor->importStylesheet() -- Import stylesheet

Descripción

class **XSLTProcessor** {

void **importStylesheet** (DOMDocument stylesheet)

}

This method import the stylesheet into the **XSLTProcessor** for transformations.

Lista de parámetros

stylesheet

The imported style sheet as a **DOMDocument** object.

Valores retornados

No value is returned.

XSLTProcessor->registerPHPFunctions()

XSLTProcessor->registerPHPFunctions() -- Enables the ability to use PHP functions as XSLT functions

Descripción

class **XSLTProcessor** {

void **registerPHPFunctions** (void)

}

This method enables the ability to use PHP functions as XSLT functions within XSL stylesheets.

Valores retornados

No value is returned.

XSLTProcessor->removeParameter()

XSLTProcessor->removeParameter() -- Remove parameter

Descripción

class **XSLTProcessor** {

bool **removeParameter** (string namespaceURI, string localName)

}

Removes a parameter, if set. This will make the processor use the default value for the parameter as specified in the stylesheet.

Lista de parámetros

namespaceURI

The namespace URI of the XSLT parameter.

localName

The local name of the XSLT parameter.

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Ver también

XSLTProcessor->setParameter()
XSLTProcessor->getParameter()

XSLTProcessor->setParameter()

XSLTProcessor->setParameter() -- Set value for a parameter

Descripción

class **XSLTProcessor** {

bool **setParameter** (string namespace, mixed name [, string value])

}

Sets the value of one or more parameters to be used in subsequent transformations with **XSLTProcessor**. If the parameter doesn't exist in the stylesheet it will be ignored.

Lista de parámetros

namespaceURI

The namespace URI of the XSLT parameter.

localName

The local name of the XSLT parameter. This can be either a string representing the parameter name or an array of *name* => *value* pairs.

value

The new value of the XSLT parameter.

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Ejemplos

Ejemplo 1. Changing the owner before the transformation

```php
<?php

$collections = array(
    'Marc Rutkowski' => 'marc',
    'Olivier Parmentier' => 'olivier'
);

$xsl = new DOMDocument;
$xsl->load('collection.xsl');

// Configure the transformer
$proc = new XSLTProcessor;
$proc->importStyleSheet($xsl); // attach the xsl rules

foreach ($collections as $name => $file) {
    // Load the XML source
    $xml = new DOMDocument;
    $xml->load('collection_' . $file . '.xml');

    $proc->setParameter('', 'owner', $name);
    $proc->transformToURI($xml, 'file:///tmp/' . $file . '.html');
}

?>
```

Ver también

XSLTProcessor->getParameter()
XSLTProcessor->removeParameter()

XSLTProcessor->transformToDoc()

XSLTProcessor->transformToDoc() -- Transform to a DOMDocument

Descripción

class **XSLTProcessor** {

DOMDocument **transformToDoc** (DOMNode doc)

}

Transforms the source node to a **DOMDocument** applying the stylesheet given by the XSLTProcessor->importStylesheet() method.

Lista de parámetros

doc

The node to be transformed.

Valores retornados

The resulting **DOMDocument** or **FALSE** on error.

Ejemplos

Ejemplo 1. Transforming to a DOMDocument

```
<?php

// Load the XML source
$xml = new DOMDocument;
$xml->load('collection.xml');

$xsl = new DOMDocument;
$xsl->load('collection.xsl');

// Configure the transformer
$proc = new XSLTProcessor;
$proc->importStyleSheet($xsl); // attach the xsl rules

echo trim($proc->transformToDoc($xml)->firstChild->wholeText);

?>
```

El resultado del ejemplo seria:

```
Hey! Welcome to Nicolas Eliaszewicz's sweet CD collection!
```

Ver también

XSLTProcessor->transformToURI()
XSLTProcessor->transformToXML()

XSLTProcessor->transformToURI()

XSLTProcessor->transformToURI() -- Transform to URI

Descripción

class **XSLTProcessor** {

int **transformToURI** (DOMDocument doc, string uri)

}

Transforms the source node to an URI applying the stylesheet given by the XSLTProcessor->importStylesheet() method.

Lista de parámetros

doc

The transformed document.

uri

Valores retornados

Returns the number of bytes written or **FALSE** if an error occurred.

Ejemplos

Ejemplo 1. Transforming to a HTML file

```
<?php

// Load the XML source
$xml = new DOMDocument;
$xml->load('collection.xml');

$xsl = new DOMDocument;
$xsl->load('collection.xsl');

// Configure the transformer
$proc = new XSLTProcessor;
$proc->importStyleSheet($xsl); // attach the xsl rules

$proc->transformToURI($xml, 'file:///tmp/out.html');

?>
```

Ver también

XSLTProcessor->transformToDoc()
XSLTProcessor->transformToXML()

XSLTProcessor->transformToXML()

XSLTProcessor->transformToXML() -- Transform to XML

Descripción

class **XSLTProcessor** {

string **transformToXML** (DOMDocument doc)

}

Transforms the source node to a string applying the stylesheet given by the XSLTProcessor->importStylesheet() method.

Lista de parámetros

doc

The transformed document.

Valores retornados

The result of the transformation as a string or **FALSE** on error.

Ejemplos

Ejemplo 1. Transforming to a string

```
<?php

// Load the XML source
$xml = new DOMDocument;
$xml->load('collection.xml');

$xsl = new DOMDocument;
$xsl->load('collection.xsl');

// Configure the transformer
$proc = new XSLTProcessor;
$proc->importStyleSheet($xsl); // attach the xsl rules

echo $proc->transformToXML($xml);

?>
```

El resultado del ejemplo seria:

```
Hey! Welcome to Nicolas Eliaszewicz's sweet CD collection!

<h1>Fight for your mind</h1><h2>by Ben Harper - 1995</h2><hr>
<h1>Electric Ladyland</h1><h2>by Jimi Hendrix - 1997</h2><hr>
```

Ver también

XSLTProcessor->transformToDoc()
XSLTProcessor->transformToURI()

CXXXVIII. XSLT functions

Aviso
Esta extensión es *EXPERIMENTAL*. Esto significa que el comportamiento de esta extensión, los nombre de sus funciones y en definitiva TODO lo documentado sobre esta extensión, puede cambiar en una futura versión de PHP SIN AVISO. La advertencia queda hecha, y utilizar esta extensión queda bajo su propia responsabilidad.

Introduction

About XSLT and Sablotron

XSLT (Extensible Stylesheet Language (XSL) Transformations) is a language for transforming XML documents into other XML documents. It is a standard defined by The World Wide Web consortium (W3C). Information about XSLT and related technologies can be found at http://www.w3.org/TR/xslt.

Installation

This extension uses Sabloton and expat, which can both be found at http://www.gingerall.com/. Binaries are provided as well as source.

On UNIX, run **configure** with the *--with-sablot* and *--enable-sablot-errors-descriptive* options. The Sablotron library should be installed somewhere your compiler can find it.

About This Extension

This PHP extension implements support Sablotron from Ginger Alliance in PHP. This toolkit lets you transform XML documents into other documents, including new XML documents, but also into HTML or other target formats. It basically provides a standardized and portable template mechanism, separating content and design of a website.

Tabla de contenidos

xslt_backend_info

(PHP 4 >= 4.3.0)

xslt_backend_info -- Returns the information on the compilation settings of the backend

Description

string **xslt_backend_info** (void)

xslt_backend_info() returns a string with information about the compilation setting of the backend or an error string when no information available.

Ver también

xslt_backend_name(), y **xslt_backend_version()**.

xslt_backend_name

(PHP 4 >= 4.3.0)

xslt_backend_name -- Returns the name of the backend

Description

string **xslt_backend_name** (void)

xslt_backend_name() will always return Sablotron.

Ejemplos

Ejemplo 1. xslt_backend_name() example

```
<?php

echo xslt_backend_name(); // Sablotron

?>
```

Ver también

xslt_backend_info(), y **xslt_backend_version()**.

xslt_backend_version

(PHP 4 >= 4.3.0)

xslt_backend_version -- Returns the version number of Sablotron

Description

string **xslt_backend_version** (void)

xslt_backend_version() returns the version number of Sablotron if available, **FALSE** otherwise.

Ejemplos

Ejemplo 1. xslt_backend_version() example

```
<?php

echo xslt_backend_version(); // 0.98 for example

?>
```

Ver también

xslt_backend_name(), y **xslt_backend_info()**.

xslt_create

(PHP 4 >= 4.0.3)

xslt_create -- Create a new XSL processor.

Description

resource **xslt_create** (void)

This function returns a handle for a new XSL processor. This handle is needed in all subsequent calls to XSL functions.

xslt_errno

(PHP 4 >= 4.0.3)

xslt_errno -- Return the current error number

Description

int **xslt_errno** ([int xh])

Return the current error number of the given XSL processor. If no handle is given, the last error number that occured anywhere is returned.

xslt_error

(PHP 4 >= 4.0.3)

xslt_error -- Return the current error string

Description

mixed **xslt_error** ([int xh])

Return the current error string of the given XSL processor. If no handle is given, the last error string that occured anywhere is returned.

xslt_free

(PHP 4 >= 4.0.3)

xslt_free -- Free XSLT processor

Description

void **xslt_free** (resource xh)

Free the XSLT processor identified by the given handle.

xslt_getopt

(PHP 4 >= 4.3.0)

xslt_getopt -- Get options on a given xsl processor

Description

int **xslt_getopt** (resource processor)

xslt_getopt() returns the options on the given *processor*.

Ver también

xslt_setopt().

xslt_process

(PHP 4 >= 4.0.3)

xslt_process -- unknown

Description

unknown **xslt_process** (unknown)

This function lacks a prototype definition.

xslt_set_base

(PHP 4 >= 4.0.5)

xslt_set_base -- Set the base URI for all XSLT transformations

Description

void **xslt_set_base** (resource xh, string uri)

Sets the base URI for all XSLT transformations, the base URI is used with Xpath instructions to resolve document() and other commands which access external resources. It is also used to resolve URIs for the <xsl:include> and <xsl:import> elements.

As of 4.3, the default base URI is the directory of the executing script. In effect, it is the directory name value of the **__FILE__** constant. Prior to 4.3, the default base URI was less predictable.

> **Nota:** Tener en cuanta que se requiere el uso de *file://* delante del PATH si se utiliza Windows.

xslt_set_encoding

(PHP 4 >= 4.0.5)

xslt_set_encoding -- Set the encoding for the parsing of XML documents

Description

void **xslt_set_encoding** (resource xh, string encoding)

Set the output encoding for the XSLT transformations. When using the Sablotron backend, this option is only available when you compile Sablotron with encoding support.

xslt_set_error_handler

(PHP 4 >= 4.0.4)

xslt_set_error_handler -- Set an error handler for a XSLT processor

Description

void **xslt_set_error_handler** (resource xh, mixed handler)

Set an error handler function for the XSLT processor given by *xh*, this function will be called whenever an error occurs in the XSLT transformation (this function is also called for notices).

The user function needs to accept four parameters: the XSLT processor, the error level, the error code and an array of messages. The function can be shown as: ***error_handler*** (resource xh, int error_level, int error_code, array messages)

Ejemplos

Ejemplo 1. xslt_set_error_handler() Example

```
<?php

// Our XSLT error handler
function xslt_error_handler($handler, $errno, $level, $info)
{
  // for now, let's just see the arguments
  var_dump(func_get_args());
}

// XML content :
$xml='<?xml version="1.0"?>
<para>
 oops, I misspelled the closing tag
</pata>';

// XSL content :
$xsl='<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
   <strong><xsl:value-of select="para"/></strong>
</xsl:template>
</xsl:stylesheet>';

$xh = xslt_create();

xslt_set_error_handler($xh, "xslt_error_handler");

echo xslt_process($xh, 'arg:/_xml', 'arg:/_xsl',
                  NULL, array("/_xml" => $xml, "/_xsl" => $xsl));

?>
```

El resultado del ejemplo seria algo similar a:

```
array(4) {
  [0]=>
  resource(1) of type (XSLT Processor)
  [1]=>
  int(3)
  [2]=>
  int(0)
  [3]=>
  array(6) {
    ["msgtype"]=>
    string(5) "error"
    ["code"]=>
    string(1) "2"
    ["module"]=>
    string(9) "Sablotron"
    ["URI"]=>
    string(9) "arg:/_xml"
    ["line"]=>
    string(1) "4"
    ["msg"]=>
    string(34) "XML parser error 7: mismatched tag"
  }
}
```

Ver también

xslt_set_object() if you want to use an object method as handler.

xslt_set_log

(PHP 4 >= 4.0.6)

xslt_set_log -- Set the log file to write log messages to

Description

void **xslt_set_log** (resource xh [, mixed log])

xh

A reference to the XSLT parser.

log

This parameter is either a boolean value which toggles logging on and off, or a string containing the logfile in which log errors too.

This function allows you to set the file in which you want XSLT log messages to, XSLT log messages are different than error messages, in that log messages are not actually error messages but rather messages related to the state of the XSLT processor. They are useful for debugging XSLT, when something goes wrong.

By default logging is disabled, in order to enable logging you must first call **xslt_set_log()** with a boolean parameter which enables logging, then if you want to set the log file to debug to, you must then pass it a string containing the filename.

Nota: Tener en cuanta que se requiere el uso de *file://* delante del PATH si se utiliza Windows.

Ejemplos

Ejemplo 1. Using the XSLT Logging features

```
<?php

$xh = xslt_create();
xslt_set_log($xh, true);
xslt_set_log($xh, getcwd() . '/myfile.log');

$result = xslt_process($xh, 'dog.xml', 'pets.xsl');
echo $result;

xslt_free($xh);
?>
```

xslt_set_object

(PHP 4 >= 4.3.0)

xslt_set_object -- Sets the object in which to resolve callback functions

Description

int **xslt_set_object** (resource processor, object &obj)

This function allows to use the *processor* inside an *object* and to resolve all callback functions in it.

The callback functions can be declared with **xml_set_sax_handlers()**, **xslt_set_scheme_handlers()** or **xslt_set_error_handler()** and are assumed to be methods of *object*.

Ejemplos

Ejemplo 1. Using your own error handler as a method

```
<?php

class my_xslt_processor {

    var $_xh; // our XSLT processor

    function my_xslt_processor()
    {
        $this->_xh = xslt_create();

        // Make $this object the callback resolver
        xslt_set_object($this->_xh, $this);

        // Let's handle the errors
        xslt_set_error_handler($this->_xh, "my_xslt_error_handler");
    }

    function my_xslt_error_handler($handler, $errno, $level, $info)
    {
        // for now, let's just see the arguments
        var_dump(func_get_args());
    }
}

?>
```

xslt_set_sax_handler

(4.0.3 - 4.0.6 only)

xslt_set_sax_handler -- Set SAX handlers for a XSLT processor

Description

bool **xslt_set_sax_handler** (resource xh, handlers)

Set SAX handlers on the ressource handle given by xh.

xslt_set_sax_handlers

(PHP 4 >= 4.0.6)

xslt_set_sax_handlers -- Set the SAX handlers to be called when the XML document gets processed

Description

void **xslt_set_sax_handlers** (resource processor, array handlers)

xslt_set_sax_handlers() registers the SAX *handlers* for the document, given a XSLT *processor*

resource.

handlers should be an array in the following format:

```
<?php

$handlers = array(

  "document" => array(
    "start_doc",
    "end_doc"),

  "element"  => array(
    "start_element",
    "end_element"),

  "namespace" => array(
    "start_namespace",
    "end_namespace"),

  "comment"   => "comment",

  "pi"        => "pi",

  "character" => "characters"

);
?>
```

Where the functions follow the syntax described for the scheme handler functions.

> **Nota:** The given array does not need to contain all of the different sax handler elements (although it can), but it only needs to conform to "handler" => "function" format described above.

Each of the individual SAX handler functions are in the format below:

- ***start_doc*** (resource processor)
- ***end_doc*** (resource processor)
- ***start_element*** (resource processor, string name, array attributes)
- ***end_element*** (resource processor, string name)
- ***start_namespace*** (resource processor, string prefix, string uri)
- ***end_namespace*** (resource processor, string prefix)
- ***comment*** (resource processor, string contents)
- ***pi*** (resource processor, string target, string contents)
- ***characters*** (resource processor, string contents)

Using **xslt_set_sax_handlers()** doesn't look very different than running a SAX parser like **xml_parse()** on the result of an **xslt_process()** transformation.

Ejemplos

Ejemplo 1. xslt_set_sax_handlers() Example

```php
<?php
// From ohlesbeauxjours at yahoo dot fr
// Here's a simple example that applies strtoupper() on
// the content of every <auteur> tag and then displays the
// resulting XML tree:

$xml='<?xml version="1.0"?>
<books>
 <book>
  <title>Mme Bovary</title>
  <author>Gustave Flaubert</author>
 </book>
 <book>
  <title>Mrs Dalloway</title>
  <author>Virginia Woolf</author>
 </book>
</books>';

$xsl='<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" encoding="ISO-8859-1" indent="no" omit-xml-declaration="yes"/
<xsl:template match="/">
 <xsl:for-each select="books/book">
  <livre>
   <auteur><xsl:value-of select="author/text()"/></auteur>
  </livre>
 </xsl:for-each>
</xsl:template>
</xsl:stylesheet>';

// Handlers :
function start_document()
{
  // start reading the document
}

function end_document()
{
  // end reading the document
}

function start_element($parser, $name, $attributes)
{
  global $result,$tag;
  $result .= "<". $name . ">";
  $tag = $name;
}

function end_element($parser, $name)
{
  global $result;
  $result .= "</" . $name . ">";
}

function characters($parser, $data)
{
  global $result,$tag;
  if ($tag == "auteur" ) {
    $data = strtoupper($data);
  }
  $result .= $data;
}

// Transformation :
$xh = xslt_create();
$handlers = array("document" => array("start_document","end_document"),
   "element" => array("start_element","end_element"),
   "character" => "characters");

xslt_set_sax_handlers($xh, $handlers);
xslt_process($xh, 'arg:/_xml', 'arg:/_xsl', NULL, array("/_xml"=>$xml, "/_xsl"=>$xsl));
xslt_free($xh);
```

You can also use **xslt_set_object()** if you want to implement your handlers in an object.

Ejemplo 2. Object oriented handler

```
<?php
// This is the object oriented version of the previous example
class data_sax_handler {

  var $buffer, $tag, $attrs;

  var $_xh;

  function data_sax_handler($xml, $xsl)
  {
      // our xslt resource
      $this->_xh = xslt_create();

      xslt_set_object($this->_xs, $this);

      // configure sax handlers
      $handlers = array(
        "document" => array('start_document', 'end_document'),
        "element" => array('start_element', 'end_element'),
        "character" => 'characters'
      );

      xslt_set_sax_handlers($this->_xh, $handlers);

      xslt_process($this->_xh, 'arg:/_xml', 'arg:/_xsl', NULL, array("/_xml"=>$xml, "/_
      xslt_free($this->_xh);


  }

  function start_document()
  {
        // start reading the document
  }

  function end_document() {
        // complete reading the document
  }

  function start_element($parser, $name, $attributes) {
        $this->tag = $name;
        $this->buffer .= "<" . $name . ">";
        $this->attrs = $attributes;
  }

  function end_element($parser, $name)
  {
        $this->tag = '';
        $this->buffer .= "</" . $name . ">";
  }

  function characters($parser, $data)
  {
    if ($this->tag == 'auteur') {
          $data = strtoupper($data);
    }
    $this->buffer .= $data;
  }

  function get_buffer() {
    return $this->buffer;
  }

}

$exec = new data_sax_handler($xml, $xsl);

?>
```

Both examples will output:

```
<livre>
   <auteur>GUSTAVE FLAUBERT</auteur>
</livre>
<livre>
   <auteur>VIRGINIA WOOLF</auteur>
</livre>
```

xslt_set_scheme_handler

(4.0.5 - 4.0.6 only)

xslt_set_scheme_handler -- Set Scheme handlers for a XSLT processor

Description

void **xslt_set_scheme_handler** (resource xh, array handlers)

Set Scheme handlers on the resource handle given by *xh*. Scheme handlers should be an array with the format (all elements are optional):

```
array(
[get_all] => get all handler,
[open] => open handler,
[get] => get handler,
[put] => put handler,
[close] => close handler
)
```

xslt_set_scheme_handlers

(PHP 4 >= 4.0.6)

xslt_set_scheme_handlers -- Set the scheme handlers for the XSLT processor

Description

void **xslt_set_scheme_handlers** (resource processor, array handlers)

Aviso
Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

xslt_setopt

(PHP 4 >= 4.3.0)

xslt_setopt -- Set options on a given xsl processor

Description

int **xslt_setopt** (resource processor, int newmask)

xslt_setopt() sets the options specified by *newmask* on the given *processor*.

newmask is a bitmask constructed with the following constants:

- **XSLT_SABOPT_PARSE_PUBLIC_ENTITIES** - Tell the processor to parse public entities. By default this has been turned off.
- **XSLT_SABOPT_DISABLE_ADDING_META** - Do not add the meta tag "Content-Type" for HTML output. The default is set during the compilation of the processor.
- **XSLT_SABOPT_DISABLE_STRIPPING** - Suppress the whitespace stripping (on data files only).
- **XSLT_SABOPT_IGNORE_DOC_NOT_FOUND** - Consider unresolved documents (the document() function) non-lethal.

Ejemplos

Ejemplo 1. xslt_setopt() Example

```
<?php

$xh = xslt_create();

// Tell Sablotron to process public entities
xslt_setopt($xh, XSLT_SABOPT_PARSE_PUBLIC_ENTITIES);

// Let's also ask him to suppress whitespace stripping
xslt_setopt($xh, xslt_getopt($xh) | XSLT_SABOPT_DISABLE_STRIPPING);

?>
```

Ver también

xslt_getopt().

CXXXIX. YAZ

The **yaz()** functions wrap the YAZ API. The home page of the project is http://www.indexdata.dk/yaz/. Information about the phpyaz module can be found at http://www.indexdata.dk/phpyaz/.

PHP/YAZ is much simpler to use than the C API for YAZ but less flexible. The intent is to make it easy to build basic client functions. It supports persistent stateless connections very similar to those offered by the various SQL APIs that are available for PHP. This means that sessions are stateless but shared amongst users, thus saving the connect and INIT steps in many cases.

Before compiling PHP with the PHP/YAZ module you'll need the YAZ toolkit. Build YAZ and install it. Build PHP with your favourite modules and add option --with-yaz. Your task is roughly the following:

```
gunzip -c yaz-1.6.tar.gz|tar xf -
gunzip -c php-4.0.X.tar.gz|tar xf -
cd yaz-1.6
./configure --prefix=/usr
make
make install
cd ../php-4.0.X
./configure --with-yaz=/usr/bin
make
make install
```

PHP/YAZ keeps track of connections with targets (Z-Associations). A positive integer represents the ID of a particular association.

The script below demonstrates the parallel searching feature of the API. When invoked it either prints a query form (if no arguments are supplied) or if there are arguments (term and one or more hosts) it searches the targets in array host.

Ejemplo 1. YAZ()

```
$num_hosts = count ($host);
if (empty($term) || count($host) == 0) {
    echo '<form method="get">
    <input type="checkbox"
    name="host[]" value="bagel.indexdata.dk/gils">
        GILS test
    <input type="checkbox"
    name="host[]" value="localhost:9999/Default">
        local test
    <input type="checkbox" checked="1"
    name="host[]" value="z3950.bell-labs.com/books">
        BELL Labs Library
    <br>
    RPN Query:
    <input type="text" size="30" name="term">
    <input type="submit" name="action" value="Search">
    ';
} else {
    echo 'You searced for ' . htmlspecialchars($term) . '<br>';
    for ($i = 0; $i > $num_hosts; $i++) {
        $id[] = yaz_connect($host[$i]);
        yaz_syntax($id[$i],"sutrs");
        yaz_search($id[$i],"rpn",$term);
    }
    yaz_wait();
    for ($i = 0; $i < $num_hosts; $i++) {
        echo '<hr>' . $host[$i] . ":";
        $error = yaz_error($id[$i]);
        if (!empty($error)) {
            echo "Error: $error";
        } else {
            $hits = yaz_hits($id[$i]);
            echo "Result Count $hits";
        }
        echo '<dl>';
        for ($p = 1; $p <= 10; $p++) {
            $rec = yaz_record($id[$i],$p,"string");
            if (empty($rec)) continue;
            echo "<dt><b>$p</b></dt><dd>";
            echo ereg_replace("\n", "<br>\n",$rec);
            echo "</dd>";
        }
        echo '</dl>';
    }
}
```

Tabla de contenidos

yaz_addinfo

(PHP 4 >= 4.0.1, PHP 5)

yaz_addinfo -- Returns additional error information

Description

int **yaz_addinfo** (int id)

Returns additional error message for target (last request). An empty string is returned if last operation was a success or if no additional information was provided by the target.

yaz_ccl_conf

(PHP 4 >= 4.0.5, PHP 5)

yaz_ccl_conf -- Configure CCL parser

Description

int **yaz_ccl_conf** (resource id, array config)

This function configures the CCL query parser for a server with definitions of access points (CCL qualifiers) and their mapping to RPN. To map a specific CCL query to RPN afterwards call the **yaz_ccl_parse()** function. Each index of the array *config* is the name of a CCL field and the corresponding value holds a string that specifies a mapping to RPN. The mapping is a sequence of attribute-type, attribute-value pairs. Attribute-type and attribute-value is separated by an equal sign

(=). Each pair is separated by white space.

Ejemplo 1. CCL configuration

In the example below, the CCL parser is configured to support three CCL fields: *ti*, *au* and *isbn*. Each field is mapped to their BIB-1 ñalent. It is assumed that variable *$id* is the connection ID.

```
<?php
$fields["ti"] = "1=4";
$fields["au"] = "1=1";
$fields["isbn"] = "1=7";
yaz_ccl_conf($id, $fields);
?>
```

yaz_ccl_parse

(PHP 4 >= 4.0.5, PHP 5)

yaz_ccl_parse -- Invoke CCL Parser

Description

bool **yaz_ccl_parse** (resource id, string query, array &result)

This function invokes a CCL parser. It converts a given CCL FIND query to an RPN query which may be passed to the **yaz_search()** function to perform a search. To define a set of valid CCL fields call **yaz_ccl_conf()** prior to this function. If the supplied *query* was successfully converted to RPN, this function returns **TRUE**, and the index *rpn* of the supplied array *result* holds a valid RPN query. If the query could not be converted (because of invalid syntax, unknown field, etc.) this function returns **FALSE** and three indexes are set in the resulting array to indicate the cause of failure: *errorcode* CCL error code (integer), *errorstring* CCL error string, and *errorpos* approximate position in query of failure (integer is character position).

Ejemplo 1. CCL Parsing

We will try to search using CCL. In the example below, *$ccl* is a CCL query.

```
<?php
yaz_ccl_conf($id, $fields);  // see example for yaz_ccl_conf
if (!yaz_ccl_parse($id, $ccl, &$cclresult)) {
    echo 'Error: ' . $cclresult["errorstring"];
} else {
    $rpn = $cclresult["rpn"];
    yaz_search($id, "rpn", $rpn);
}
?>
```

yaz_close

(PHP 4 >= 4.0.1, PHP 5)

yaz_close -- Closes a YAZ connection

Description

int **yaz_close** (int id)

Closes a connection to a target. The application can no longer refer to the target with the given id.

yaz_connect

(PHP 4 >= 4.0.1, PHP 5)

yaz_connect -- Returns a positive association ID on success; zero on failure

Description

int **yaz_connect** (string zurl)

yaz_connect() prepares for a connection to a Z39.50 target. The zurl argument takes the form host [:port][/database]. If port is omitted 210 is used. If database is omitted Default is used. This function is non-blocking and doesn't attempt to establish a socket - it merely prepares a connect to be performed later when **yaz_wait()** is called.

yaz_database

(PHP 4 >= 4.0.6, PHP 5)

yaz_database -- Specifies the databases within a session

Description

bool **yaz_database** (resource id, string databases)

This function specifies one or more databases to be used in search, retrieval, etc. - overriding databases specified in call to **yaz_connect()**. Multiple databases are separated by a plus sign +.

This function allows you to change databases within a session.

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

yaz_element

(PHP 4 >= 4.0.1, PHP 5)

yaz_element -- Specifies Element-Set Name for retrieval

Description

bool **yaz_element** (resource id, string elementset)

This function sets the element set name for retrieval. Call this function before **yaz_search()** or

yaz_present() to specify the element set name for records to be retrieved. Most servers support *F* (for full records) and *B* (for brief records).

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

yaz_errno

(PHP 4 >= 4.0.1, PHP 5)

yaz_errno -- Returns error number

Description

int **yaz_errno** (int id)

Returns error for target (last request). A positive value is returned if the target returned a diagnostic code; a value of zero is returned if no errors occurred (success); negative value is returned for other errors targets didn't indicate the error in question.

yaz_errno() should be called after network activity for each target - (after yaz_wait() returns) to determine the success or failure of the last operation (e.g. search).

yaz_error

(PHP 4 >= 4.0.1, PHP 5)

yaz_error -- Returns error description

Description

int **yaz_error** (int id)

Returns error message for target (last request). An empty string is returned if last operation was a success.

yaz_error() returns a english message corresponding to the last error number as returned by yaz_errno().

yaz_es_result

(PHP 4 >= 4.2.0, PHP 5)

yaz_es_result -- Inspects Extended Services Result

Description

array **yaz_es_result** (resource id)

Aviso

Esta función no está documentada actualmente, solamente se encuentra disponible la lista de parámetros.

yaz_get_option

(PHP 5)

yaz_get_option -- Returns value of option for connection

Description

string **yaz_get_option** (resource id, string name)

Returns the value of the option specified with *name*. If an option is not set, an empty string is returned.

See the description of **yaz_set_option()** for available options.

yaz_hits

(PHP 4 >= 4.0.1, PHP 5)

yaz_hits -- Returns number of hits for last search

Description

int **yaz_hits** (int id)

yaz_hits() returns number of hits for last search.

yaz_itemorder

(PHP 4 >= 4.0.5, PHP 5)

yaz_itemorder -- Prepares for Z39.50 Item Order with an ILL-Request package

Description

int **yaz_itemorder** (resource id, array args)

This function prepares for an Extended Services request using the Profile for the Use of Z39.50 Item Order Extended Service to Transport ILL (Profile/1). See this and the specification. The *args* parameter must be a hash array with information about the Item Order request to be sent. The key of the hash is the name of the corresponding ASN.1 tag path. For example, the ISBN below the Item-ID has the key item-id,ISBN.

The ILL-Request parameters are:

protocol-version-num
transaction-id,initial-requester-id,person-or-institution-symbol,person
transaction-id,initial-requester-id,person-or-institution-symbol,institution
transaction-id,initial-requester-id,name-of-person-or-institution,name-of-person
transaction-id,initial-requester-id,name-of-person-or-institution,name-of-institution
transaction-id,transaction-group-qualifier
transaction-id,transaction-qualifier
transaction-id,sub-transaction-qualifier
service-date-time,this,date
service-date-time,this,time
service-date-time,original,date
service-date-time,original,time
requester-id,person-or-institution-symbol,person
requester-id,person-or-institution-symbol,institution
requester-id,name-of-person-or-institution,name-of-person
requester-id,name-of-person-or-institution,name-of-institution
responder-id,person-or-institution-symbol,person
responder-id,person-or-institution-symbol,institution
responder-id,name-of-person-or-institution,name-of-person
responder-id,name-of-person-or-institution,name-of-institution
transaction-type
delivery-address,postal-address,name-of-person-or-institution,name-of-person
delivery-address,postal-address,name-of-person-or-institution,name-of-institution
delivery-address,postal-address,extended-postal-delivery-address
delivery-address,postal-address,street-and-number
delivery-address,postal-address,post-office-box
delivery-address,postal-address,city
delivery-address,postal-address,region
delivery-address,postal-address,country
delivery-address,postal-address,postal-code
delivery-address,electronic-address,telecom-service-identifier
delivery-address,electronic-address,telecom-service-addreess
billing-address,postal-address,name-of-person-or-institution,name-of-person
billing-address,postal-address,name-of-person-or-institution,name-of-institution
billing-address,postal-address,extended-postal-delivery-address
billing-address,postal-address,street-and-number
billing-address,postal-address,post-office-box
billing-address,postal-address,city
billing-address,postal-address,region
billing-address,postal-address,country
billing-address,postal-address,postal-code
billing-address,electronic-address,telecom-service-identifier
billing-address,electronic-address,telecom-service-addreess
ill-service-type
requester-optional-messages,can-send-RECEIVED
requester-optional-messages,can-send-RETURNED
requester-optional-messages,requester-SHIPPED
requester-optional-messages,requester-CHECKED-IN
search-type,level-of-service
search-type,need-before-date
search-type,expiry-date
search-type,expiry-flag
place-on-hold
client-id,client-name
client-id,client-status

client-id,client-identifier
item-id,item-type
item-id,call-number
item-id,author
item-id,title
item-id,sub-title
item-id,sponsoring-body
item-id,place-of-publication
item-id,publisher
item-id,series-title-number
item-id,volume-issue
item-id,edition
item-id,publication-date
item-id,publication-date-of-component
item-id,author-of-article
item-id,title-of-article
item-id,pagination
item-id,ISBN
item-id,ISSN
item-id,additional-no-letters
item-id,verification-reference-source
copyright-complicance
retry-flag
forward-flag
requester-note
forward-note

There are also a few parameters that are part of the Extended Services Request package and the ItemOrder package:

package-name
user-id
contact-name
contact-phone
contact-email
itemorder-item

yaz_present

(PHP 4 >= 4.0.5, PHP 5)

yaz_present -- Prepares for retrieval (Z39.50 present)

Description

bool **yaz_present** (resource id)

This function prepares for retrieval of records after a successful search. The **yaz_range()** should be called prior to this function to specify the range of records to be retrieved.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

yaz_range

(PHP 4 >= 4.0.1, PHP 5)

yaz_range -- Specifies the maximum number of records to retrieve

Description

int **yaz_range** (int id, int start, int number)

This function is used in conjunction with **yaz_search()** to specify the maximum number of records to retrieve (number) and the first record position (start). If this function is not invoked (only **yaz_search()**) start is set to 1 and number is set to 10.

Returns **TRUE** on success; **FALSE** on error.

yaz_record

(PHP 4 >= 4.0.1, PHP 5)

yaz_record -- Returns a record

Description

int **yaz_record** (int id, int pos, string type)

Returns record at position or empty string if no record exists at given position.

The **yaz_record()** function inspects a record in the current result set at the position specified. If no database record exists at the given position an empty string is returned. The argument, type, specifies the form of the returned record. If type is "string" the record is returned in a string representation suitable for printing (for XML and SUTRS). If type is "array" the record is returned as an array representation (for structured records).

yaz_scan_result

(PHP 4 >= 4.0.5, PHP 5)

yaz_scan_result -- Returns Scan Response result

Description

array **yaz_scan_result** (resource id [, array &result])

yaz_scan_result() returns terms and associated information as received from the server in the last performed **yaz_scan()**. This function returns an array (0..n-1) where n is the number of terms returned. Each value is a pair where the first item is the term, and the second item is the result-

count. If the optional parameter *result* is given it will be modified to hold additional information taken from the Scan Response: *number* (number of entries returned), *stepsize* (Step-size), *position* (position of term), *status* (Scan Status).

yaz_scan

(PHP 4 >= 4.0.5, PHP 5)

yaz_scan -- Prepares for a scan

Description

int **yaz_scan** (resource id, string type, string startterm [, array flags])

This function prepares for a Z39.50 Scan Request, where parameter *id* specifies connection. Starting term point for the scan is given by *startterm*. The form in which the starting term is specified is given by parameter *type*. Currently only type *rpn* is supported. The optional parameter *flags* specifies additional information to control the behaviour of the scan request. Three indexes are currently read from the flags: *number* (number of terms requested), *position* (preferred position of term) and *stepSize* (preferred step size). To actually transfer the Scan Request to the server and receive the Scan Response, **yaz_wait()** must be called. Upon completion of **yaz_wait()** call **yaz_error()** and **yaz_scan_result()** to handle the response.

The syntax of *startterm* is similar to the RPN query as described in **yaz_search()**. The startterm consists of zero or more *@attr*-operator specifications, then followed by exactly one token.

Ejemplo 1. PHP function that scans titles

```
<?php
function scan_titles($id, $startterm)
{
  yaz_scan($id, "rpn", "@attr 1=4 " . $startterm);
  yaz_wait();
  $errno = yaz_errno($id);
  if ($errno == 0) {
    $ar = yaz_scan_result($id, &$options);
    echo 'Scan ok; ';
    while (list($key, $val) = each($options)) {
      echo "$key = $val  ";
    }
    echo '<br /><table>';
    while (list($key, list($k, $term, $tcount)) = each($ar)) {
      if (empty($k)) continue;
      echo "<tr><td>$term</td><td>$tcount</td></tr>";
    }
    echo '</table>';
  } else {
    echo "Scan failed. Error: " . yaz_error($id) . "<br />";
  }
}
?>
```

yaz_schema

(PHP 4 >= 4.2.0, PHP 5)

yaz_schema -- Specifies schema for retrieval

Description

int **yaz_schema** (resource id, string schema)

The schema must be specified as an OID (Object Identifier) in a raw dot-notation (like *1.2.840.10003.13.4*) or as one of the known registered schemas: *GILS-schema*, *Holdings*, *Zthes*, ... This function should be called before **yaz_search()** or **yaz_present()**.

yaz_search

(PHP 4 >= 4.0.1, PHP 5)

yaz_search -- Prepares for a search

Description

int **yaz_search** (int id, string type, string query)

yaz_search() prepares for a search on the target with given id. The type represents the query type - only "rpn" is supported now in which case the third argument is a prefix notation query as used by YAZ. Like **yaz_connect()** this function is non-blocking and only prepares for a search to be executed later when **yaz_wait()** is called.

yaz_set_option

(PHP 5)

yaz_set_option -- Sets one or more options for connection

Description

string **yaz_set_option** (resource id, string name, string value)

string **yaz_set_option** (resource id, array options)

Sets option *name* to *value*.

Tabla 1. PYP/YAZ Connection Options

Name	Description
implementationName	implementation name of server
implementationVersion	implementation version of server
implementationId	implementation ID of server

Name	Description
schema	schema for retrieval. By default, no schema is used. Setting this option is ñalent to using function **yaz_schema()**
preferredRecordSyntax	record syntax for retrieval. By default, no syntax is used. Setting this option is ñalent to using function **yaz_syntax()**
start	offset for first record to be retrieved via **yaz_search()** or **yaz_present()**. First record is numbered has a start value of 0. Second record has start value 1. Setting this option in combination with option *count* has the same effect as calling **yaz_range()** except that records are numbered from 1 in **yaz_range()**
count	maximum number of records to be retrieved via **yaz_search()** or **yaz_present()**.

Name	Description
elementSetName	element-set-name for retrieval. Setting this option is ñalent to calling **yaz_element()**.

yaz_sort

(PHP 4 >= 4.1.0, PHP 5)

yaz_sort -- Sets sorting criteria

Description

int **yaz_sort** (resource id, string criteria)

This function sets sorting criteria and enables Z39.50 Sort. Call this function *before* **yaz_search()**. Using this function alone does not have any effect. When used in conjunction with **yaz_search()**, a Z39.50 Sort will be sent after a search response has been received and before any records are retrieved with Z39.50 Present (**yaz_present()**. The parameter *criteria* takes the form

field1 flags1 field2 flags2 ...

where field1 specifies the primary attributes for sort, field2 seconds, etc.. The field specifies either a numerical attribute combinations consisting of type=value pairs separated by comma (e.g. *1=4,2=1*) ; or the field may specify a plain string criteria (e.g. *title*. The flags is a sequence of the following characters which may not be separated by any white space.

Sort Flags

a

Sort ascending

d

Sort descending

i

Case insensitive sorting

s

Case sensitive sorting

Ejemplo 1. Sort Criterias

To sort on Bib1 attribute title, case insensitive, and ascending you would use the following sort criteria:

```
1=4 ia
```

If the secondary sorting criteria should be author, case sensitive and ascending you would use:

```
1=4 ia 1=1003 sa
```

yaz_syntax

(PHP 4 >= 4.0.1, PHP 5)

yaz_syntax -- Specifies the preferred record syntax for retrieval

Description

int **yaz_syntax** (int id, string syntax)

This function is used in conjunction with **yaz_search()** to specify the preferred record syntax for retrieval.

yaz_wait

(PHP 4 >= 4.0.1, PHP 5)

yaz_wait -- Executes queries

Description

int **yaz_wait** (int id, string syntax)

This function carries out networked (blocked) activity for outstanding requests which have been prepared by the functions **yaz_connect()**, **yaz_search()**. **yaz_wait()** returns when all targets have either completed all requests or otherwise completed (in case of errors).

CXL. Funciones de manejo de archivos Zip (sólo lectura)

Introducción

Este módulo permite leer de forma transparente archivos comprimidos en formato Zip y acceder a su contenido.

Requirimientos

Este módulo utiliza las funciones de la biblioteca ZZIPlib, creada por Guido Draheim. Se requiere

una versión de ZZIPlib >= 0.10.6.

Debe tenerse en cuenta que la anterior biblioteca solamente proporciona un conjunto limitado de funciones para el tratamiento de archivos en formato Zip. Para poder crear los archivos en formato Zip se requiere del uso de alguna herramienta externa a PHP.

Instalación

Para poder usar las funciones de manejo de archivos en formato Zip, se debe añadir el parámetro *--with-zip[=DIR]* a las opciones de configuración de PHP.

> **Nota:** El funcionamiento de las funciones de manejo de archivos en formato Zip es experimental en las versiones de PHP anteriores a la 4.1.0. En esta sección del manual se muestran las funciones que existen en las versiones posteriores a la 4.1.0

Configuración en tiempo de ejecución

Esta extensión no tiene directivas de configuración en `php.ini`.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Esta extensión no tiene ninguna constante definida.

Ejemplos

En el siguiente ejemplo se abre un archivo en formato Zip, se lee cada uno de los archivos contenidos en el y se muestran sus contenidos. El archivo `test2.zip` que se utiliza en el ejemplo es uno de los archivos de prueba que se incluyen en la distribución del código fuente de la libería ZZIPlib.

Ejemplo 1. Ejemplo de utilización de las funciones para manejo de archivos en formato Zip

```php
<?php

$zip = zip_open("/tmp/test2.zip");

if ($zip) {

    while ($zip_entry = zip_read($zip)) {
        echo "Nombre:                " . zip_entry_name($zip_entry) . "\n";
        echo "Tamanio sin comprimir: " . zip_entry_filesize($zip_entry) . "\n";
        echo "Tamanio comprimido:    " . zip_entry_compressedsize($zip_entry) . "\n";
        echo "Metodo de compresion:  " . zip_entry_compressionmethod($zip_entry) . "\n"

        if (zip_entry_open($zip, $zip_entry, "r")) {
            echo "Contenidos del archivo:\n";
            $buf = zip_entry_read($zip_entry, zip_entry_filesize($zip_entry));
            echo "$buf\n";

            zip_entry_close($zip_entry);
        }
        echo "\n";

    }

    zip_close($zip);

}

?>
```

Tabla de contenidos

zip_close

(PHP 4 >= 4.1.0)

zip_close -- Cierra un archivo en formato Zip

Descripción

void **zip_close** (resource zip)

Cierra un archivo en formato Zip. El parámetro *zip* debe ser un archivo en formato Zip abierto previamente con la función **zip_open()**.

Esta función no devuelve ningú valor.

Ver también **zip_open()** y **zip_read()**.

zip_entry_close

(PHP 4 >= 4.1.0)

zip_entry_close -- Cierra una entrada de directorio

Descripción

void **zip_entry_close** (resource zip_entry)

Cierra la entrada de directorio indicada en el parámetro *zip_entry*. Este parámetro *zip_entry* debe ser una entrada de directorio abierta mediante la funció **zip_entry_open()**.

Esta función no devuelve ningú valor.

Ver también **zip_entry_open()** y **zip_entry_read()**.

zip_entry_compressedsize

(PHP 4 >= 4.1.0)

zip_entry_compressedsize -- Obtiene el tamaño comprimido de una entrada de directorio

Descripción

int **zip_entry_compressedsize** (resource zip_entry)

Devuelve el tamaño comprimido de la entrada de directorio indicada en el parámetro *zip_entry*. Este parámetro *zip_entry* debe ser una entrada de directorio válida obtenida mediante la función **zip_read()**.

Ver también **zip_open()** y **zip_read()**.

zip_entry_compressionmethod

(PHP 4 >= 4.1.0)

zip_entry_compressionmethod -- Obtiene el método de compresión utilizado por una entrada de directorio

Descripción

string **zip_entry_compressionmethod** (resource zip_entry)

Devuelve el método de compresión de la entrada de directorio especificada en el parámetro *zip_entry*. El parámetro *zip_entry* debe ser una entrada de directorio válida obtenida mediante la función **zip_read()**.

Ver tamién **zip_open()** y **zip_read()**.

zip_entry_filesize

(PHP 4 >= 4.1.0)

zip_entry_filesize -- Obtiene el tamaño real de una entrada de directorio

Descripción

int **zip_entry_filesize** (resource zip_entry)

Devuelve el tamaño real (no comprimido) de la entrada de directorio indicada en el parámetro *zip_entry*. Este parámetro *zip_entry* debe ser una entrada de directorio válida obtenida mediante la función **zip_read()**.

Ver también **zip_open()** y **zip_read()**.

zip_entry_name

(PHP 4 >= 4.1.0)

zip_entry_name -- Obtiene el nombre de una entrada de directorio

Descripción

string **zip_entry_name** (resource zip_entry)

Devuelve el nombre de la entrada de directorio indicada en el parámetro *zip_entry*. El parámetro *zip_entry* debe ser una entrada de directorio válida obtenida mediante la función **zip_read()**.

Ver también **zip_open()** y **zip_read()**.

zip_entry_open

(PHP 4 >= 4.1.0)

zip_entry_open -- Abre una entrada de directorio en modo solo lectura

Descripción

bool **zip_entry_open** (resource zip, resource zip_entry [, string modo])

Abre una entrada de directorio de un archivo en formato zip para su lectura. El parámetro *zip* es un apuntador válido devuelto por la función **zip_open()**. El parámetro *zip_entry* es una entrada de directorio válida obtenida mediante la función **zip_read()**. El parámetro opcional *modo* puede ser cualquiera de los modos descritos en la documentación de la función **fopen()**.

> **Nota:** Actualmente, el parámetro *modo* no se tiene en cuenta y siempre se considera que es *"rb"*. La razón de este comportamiento es que, de momento, PHP solo soporta el formato Zip en operaciones de lectura y no de escritura. En la documentación de la

función **fopen()** se explican todos los modos disponibles, incluido el modo *"rb"*.

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

> **Nota:** Al contrario de lo que sucede con la función **fopen()** y otras similares, el valor que devuelve **zip_entry_open()** solamente indica el resultado de la operación efectuada y no es necesario para realizar las funciones de lectura y cierre de esa entrada de directorio.

Ver también **zip_entry_read()** y **zip_entry_close()**.

zip_entry_read

(PHP 4 >= 4.1.0)

zip_entry_read -- Lee los datos de una entrada de directorio abierta

Descripción

string **zip_entry_read** (resource zip_entry [, int longitud])

Lee como máximo un número de bytes igual al parámetro *longitud* en una entrada de directorio abierta. Si no se especifica el parámetro *longitud*, la función **zip_entry_read()** intentará leer 1024 bytes. El parámetro *zip_entry* es una entrada de directorio válida obtenida mediante la función **zip_read()**.

> **Nota:** El parámetro *longitud* indica el número de bytes que se quieren leer del archivo una vez descomprimido.

La función devuelve los datos leidos o **FALSE** si se ha llegado al final del archivo.

Ver también **zip_entry_open()**, **zip_entry_close()** y **zip_entry_filesize()**.

zip_open

(PHP 4 >= 4.1.0)

zip_open -- Abre un archivo en formato Zip

Descripción

resource **zip_open** (string nombre_archivo)

Abre un archivo en formato Zip para poder leerlo. El parámetro *nombre_archivo* es el nombre del archivo en formato Zip que se quiere abrir.

La función devuelve un apuntador que se utiliza con las funciones **zip_read()** y **zip_close()** o devuelve **FALSE** si no existe el archivo *nombre_archivo*.

Ver también **zip_read()** y **zip_close()**.

zip_read

(PHP 4 >= 4.1.0)

zip_read -- Lee la siguiente entrada de un archivo en formato Zip

Descripción

resource **zip_read** (resource zip)

Lee la siguiente entrada de un archivo en formato Zip. El parámetro *zip* debe ser un apuntador válido obtenido mediante la función **zip_open()**.

La función devuelve una entrada de directorio válida para su uso con cualquiera de las funciones **zip_entry_...()** o **FALSE** si ya no hay mas entradas para leer en el archivo en formato Zip.

Ver también **zip_open()**, **zip_close()**, **zip_entry_open()** y **zip_entry_read()**.

CXLI. Funciones de Compresión Zlib

Introducción

Este módulo le permite leer y escribir de forma transparente sobre archivos gzip (.gz) comprimidos, a través de algunas versiones de la mayoría de funciones del sistema de archivos que trabajen con archivos comprimidos-gzip (y archivos sin comprimir también, pero no con sockets).

> **Nota:** La versión 4.0.4 introdujo una envoltura fopen para archivos-.gz, de modo que puede usar una URL especial 'zlib:' para acceder a archivos comprimidos de forma transparente usando las funciones de acceso normales f*() si precede el nombre o ruta de archivo con un prefijo 'zlib:' al llamar a **fopen()**.
>
> En la versión 4.3.0, este prefijo especial ha cambiado a 'zlib://' para prevenir ambigüedades con nombres de archivo que contengan ':'.
>
> Esta característica requiere una biblioteca C de tiempo de ejecución que provee la función *fopencookie()*. Hasta donde se conoce, la biblioteca de C GNU es la única que ofrece esta característica.

Requirimientos

Este módulo usa las funciones de zlib, por Jean-loup Gailly y Mark Adler. Debe usar una versión de zlib >= 1.0.9 con éste módulo.

Instalación

Soporte Zlib en PHP no está activo por defecto. Usted necesitará configurar PHP *--with-zlib[=DIR]*

La versión para Windows de *PHP* tiene soporte nativo para esta extensión. No se necesita cargar ninguna extensión adicional para usar estas funciones.

> **Nota:** soporte integrado para zlib en Windows está disponible con PHP 4.3.0

Configuración en tiempo de ejecución

El comportamiento de estas funciones está afectado por los valores definidos en `php.ini`.

La extensión de zlib ofrece la opción de comprimir transparentemente tus páginas web al vuelo, si el navegador de internet soporta esto. Por lo tanto hay tres opciones en el archivo de configuración `php.ini`.

Tabla 1. Opciones de configuración Zlib

Nombre	Valor por Defecto	Modificable
zlib.output_compression	"Off"	PHP_INI_ALL
zlib.output_compression_level	"-1"	PHP_INI_ALL
zlib.output_handler	""	PHP_INI_ALL

Para más detalles y definiciones de las constantes PHP_INI_* vea **ini_set()**.

A continuación se presenta una corta explicación de las directivas de configuración.

zlib.output_compression **boolean**/**integer**

Si se desea comprimir páginas transparentemente. Si esta opción está en "On" en `php.ini` o en la configuración del Apache, las páginas son comprimidas si el navegador envía un encabezado "Accept-Encoding: gzip" o "deflate". "Content-Encoding: gzip" (respectivamente "deflate") y "Vary: Accept-Encoding", los encabezados son agregados a la salida.

Esta opción también acepta valores enteros en vez de los boleanos "On"/"Off", usando esto usted puede fijar el tamaño del buffer de salida (default es 4KB).

> **Nota:** output_handler debe estar vacío si está puesto en 'On' en lugar de este usted debe usar zlib.output_handler.

zlib.output_compression_level entero

El nivel de compresión usado para la salida transparentemente comprimida.

zlib.output_handler cadena

Usted no puede especificar manejadores adicionales de salida si zlib.output_compression está activada. Estos ajustes no son lo mismo que output_handler sino un diferente orden.

Tipos de recursos

Esta extensión no tiene ningún tipo de recurso definido.

Constantes predefinidas

Estas constantes están definidas por esta extensión y estarán disponibles solamente cuando la extensión ha sido o bien compilada dentro de PHP o grabada dinámicamente en tiempo de ejecución.

`FORCE_GZIP` (entero)

`FORCE_DEFLATE` (entero)

Ejemplos

Este ejemplo abre un archivo temporal y escribe una cadena de prueba en él, luego imprime el contenido de este archivo dos veces.

Ejemplo 1. Pequeño Ejemplo de Zlib

```
<?php

$nombre_archivo = tempnam('/tmp', 'zlibtest') . '.gz';
echo "<html>\n<head></head>\n<body>\n<pre>\n";
$s = "&iexcl;Tan solo una prueba, prueba, prueba, prueba, prueba, prueba!\n";

// abrir el archivo para escritura con maxima compresion
$zp = gzopen($nombre_archivo, "w9");

// escribir la cadena en el archivo
gzwrite($zp, $s);

// cerrar el archivo
gzclose($zp);

// abrir el archivo para lectura
$zp = gzopen($nombre_archivo, "r");

// leer 3 caracteres
echo gzread($zp, 3);

// mostrar la salida hasta el final de archivo y cerrarlo.
gzpassthru($zp);
gzclose($zp);

echo "\n";

// abrir el archivo e imprimir su contenido (por segunda vez).
if (readgzfile($nombre_archivo) != strlen($s)) {
        echo "&iexcl;Error con las funciones zlib!";
}
unlink($nombre_archivo);
echo "</pre>\n</body>\n</html>\n";

?>
```

Tabla de contenidos

gzcompress -- Comprime una cadena
gzdeflate -- Comprime una cadena
gzencode -- Crea una cadena comprimida con gzip
gzeof -- Prueba de apuntador para el fin de archivo gz
gzfile -- Lee un archivo gz completo en una matriz
gzgetc -- Obtiene un caracter del archivo GZ apuntado
gzgets -- Obtiene una línea del archivo apuntado
gzgetss -- Obtiene una línea del archivo apuntado y le retira las etiquetas HTML
gzinflate -- Descomprime una cadena comprimida
gzopen -- Abrir un archivo gz
gzpassthru -- Imprimir todos los datos que restan en un apuntador a archivo gz
gzputs -- Alias de **gzwrite()**
gzread -- Lectura segura de archivo binario gz
gzrewind -- Reinicia la posición de un apuntador de archivo gz
gzseek -- Busca en el archivo gz apuntado
gztell -- Indica la posición de lectura/escritura del apuntador a archivo gz
gzuncompress -- Descomprime una cadena comprimida
gzwrite -- Escritura en un archivo gz, segura con material binario
readgzfile -- Imprimir un archivo gz
zlib_get_coding_type -- Regresa el tipo de codificación usada para la salida de la compresión

gzclose

(PHP 3, PHP 4 , PHP 5)

gzclose -- Cierra un apuntador de un fichero gz abierto

Descripción

bool **gzclose** (resource zp)

Cierra el apuntador del fichero gz dado.

Lista de parámetros

zp

El apuntador al archivo gz. Debe ser válido, y debe apuntar a un fichero abierto exitosamente por **gzopen()**.

Valores retornados

Devuelve **`TRUE`** si todo se llevó a cabo correctamente, **`FALSE`** en caso de fallo.

Ver también

gzopen()

gzcompress

(PHP 4 >= 4.0.1, PHP 5)

gzcompress -- Comprime una cadena

Descripción

string **gzcompress** (string datos [, int nivel])

Esta función comprime la cadena dada usando el formato de datos *ZLIB*.

Para detalles sobre el algoritmo de compresión de ZLIB, vea el documento "Espefcificación del formato de compresión de ZLIB versión 3.3" (RFC 1950).

> **Nota:** Esto *no* es lo mismo que la compresión gzip, la cuál incluye algunos encabezados. Vea **gzencode()** para compresión de gzip.

Lista de parámetros

datos

> Los datos a comprimir.

nivel

> El nivel de compresión. Puede ser dado como 0 para no compresión hasta 9 para la compresión máxima.

Valores retornados

La cadena comprimida o **`FALSE`** si ocurre un error.

Ver también

gzdeflate()
gzinflate()
gzuncompress()
gzencode()

gzdeflate

(PHP 4 >= 4.0.4, PHP 5)

gzdeflate -- Comprime una cadena

Descripción

string **gzdeflate** (string datos [, int nivel])

Esta función comprime los datos datos usando el formato de datos *DEFLATE*.

Para detalles sobre el algoritmo de compresión de DEFLATE vea el documento "Especificación del formato de compresión de datos DEFLATE versión 1.3" (RFC 1951).

Lista de parámetros

datos

Los datos a comprimir.

nivel

El nivel de compresión. Puede ser dado como 0 para no compresión hasta 9 para la compresión máxima. Si no se dá, el nivel por defecto de compresión será el valor por defecto de la libreria zlib.

Valores retornados

La cadena comprimida o **`FALSE`** si ocurre un error.

Ver también

gzinflate()
gzcompress()
gzuncompress()
gzencode()

gzencode

(PHP 4 >= 4.0.4, PHP 5)

gzencode -- Crea una cadena comprimida con gzip

Descripción

string **gzencode** (string datos [, int nivel [, int encoding_mode]])

Esta función regresa una versión comprimida de los *datos* compatible con la salida de el programa **gzip**.

Para más información sobre el formato de ficheros GZIP, vea el documento: Especificación del formato de ficheros GZIP versión 4.3 (RFC 1952).

Lista de parámetros

datos

Los datos a codificar.

nivel

El nivel de compresión. Puede ser dado como 0 para no compresión hasta 9 para la compresión máxima. Si no se dá, el nivel por defecto de compresión será el valor por defecto de la libreria zlib.

encoding_mode

El modo de codificado. Puede ser **FORCE_GZIP** (por defecto) o **FORCE_DEFLATE**.

Si usa **FORCE_DEFLATE**, obtiene una cadena descomprimida estándar de zlib (incluyendo los encabezados zlib) después de los encabezados del archivo gzip pero sin chequeo de redundancia cíclica crc32.

Valores retornados

La cadena codificada o **FALSE** en case de error.

Ejemplos

Los datos resultantes contienen los encabezados apropiados y la estructura de datos para ser un fichero .gz estádar ej.:

Ejemplo 1. Creando un fichero gzip

```
<?php
$data = implode("", file("bigfile.txt"));
$gzdata = gzencode($data, 9);
$fp = fopen("bigfile.txt.gz", "w");
fwrite($fp, $gzdata);
fclose($fp);
?>
```

Registro de cambios

Versión	Descripción
4.2	Fue agregado *nivel*. Antes **gzencode()** solo tenia los parámetros opcionales *datos* y *encoding_mode*.

Ver también

gzdeflate()
gzinflate()
gzuncompress()
gzcompress()

gzeof

(PHP 3, PHP 4 , PHP 5)

gzeof -- Prueba de apuntador para el fin de archivo gz

Descripción

int **gzeof** (resource zp)

Prueba el apuntador de archivo GZ dado en busca del EOF (fin de archivo).

Lista de parámetros

zp

El apuntador al archivo gz. Debe ser válido, y debe apuntar a un fichero abierto exitosamente por **gzopen()**.

Valores retornados

Regresa **TRUE** si el apuntador GZ está al final del archivo o si ocurrió un error, **FALSE** en caso contrario.

gzfile

(PHP 3, PHP 4 , PHP 5)

gzfile -- Lee un archivo gz completo en una matriz

Descripción

array **gzfile** (string nombre_archivo [, int usar_ruta_inclusion])

Función idéntica a **readgzfile()**, excepto que **gzfile()** devuelve el archivo en una matriz.

Lista de parámetros

nombre_archivo

El nombre del archivo.

usar_ruta_inclusion

Puedes asignar este parámetro opcional a *1*, si deseas buscar también el archivo eninclude_path.

Valores retornados

Una matriz que contiene el archivo, una línea por celda.

Ver también

readgzfile()
gzopen()

gzgetc

(PHP 3, PHP 4 , PHP 5)

gzgetc -- Obtiene un caracter del archivo GZ apuntado

Descripción

string **gzgetc** (resource zp)

Regresa una cadena conteniendo un sólo caracter (sin compresión) leído de archivo gz apuntado.

Lista de parámetros

zp

El apuntador al archivo gz. Este debe ser válido, y debe apuntar a un archivo abierto exitosamente por **gzopen()**.

Valores retornados

El caracter sin compresión o **`FALSE`** en caso de EOF (a diferencia de **gzeof()**).

Ver también

gzopen()
gzgets()

gzgets

(PHP 3, PHP 4 , PHP 5)

gzgets -- Obtiene una línea del archivo apuntado

Descripción

string **gzgets** (resource zp, int tamaño)

Obtiene a cadena (sin comprimir) de hasta tamaño -1 bytes leídos de el archivo apuntado. La lectura termina cuando tamaño -1 bytes han sido leídos, en un salto de línea o cuando se alcance el fin del archivo (EOF), lo que ocurra primero.

Lista de parámetros

zp

El apuntador al archivo gz. Debe ser válido, y debe apuntar a un fichero abierto exitosamente por **gzopen()**.

tamaño

El tamaño de los datos a obtener.

Valores retornados

La cadena sin comprimir, o **FALSE** en error.

Ver también

gzopen()
gzgetc()
gzgets()

gzgetss

(PHP 3, PHP 4 , PHP 5)

gzgetss -- Obtiene una línea del archivo apuntado y le retira las etiquetas HTML

Descripción

string **gzgetss** (resource zp, int longitud [, string etiqueta_permitida])

Identica a **gzgets()**, excepto que **gzgetss()** intenta retirar cualquier etiqueta HTML y PHP del texto que lee.

Lista de parámetros

zp

El apuntador de archivo gz. Debe ser válido, y debe apuntar a un archivo exitosamente abierto por **gzopen()**.

longitud

La longitud de los datos a obtener.

etiqueta_permitida

Puede usar este parámetro opcional para especificar que etiquetas no será retiradas.

Valores retornados

Los datos sin compresión ni etiquetas, o **FALSE** en caso de error.

Registro de cambios

Versión	Descripción
3.0.13 y 4.0b3	*etiqueta_permitida* fue agregada.

Ver también

gzopen()
gzgets()
strip_tags()

gzinflate

(PHP 4 >= 4.0.4, PHP 5)

gzinflate -- Descomprime una cadena comprimida

Descripción

string **gzinflate** (string datos [, int longitud])

Está función descomprime una cadena comprimida.

Lista de parámetros

datos

Los datos comprimidos por **gzdeflate()**.

longitud

La longitud máxima de los datos a descodificar o descomprimir.

Valores retornados

The original uncompressed data or **`FALSE`** on error.

La función regresará un error si los datos ya sin comprimir son mayor a 32768 veces la longitud de los dados en el parámetro *datos* o mas que el parámetro opcional *longitud*.

Ver también

gzdeflate()
gzcompress()
gzuncompress()
gzencode()

gzopen

(PHP 3, PHP 4 , PHP 5)

gzopen -- Abrir un archivo gz

Descripción

resource **gzopen** (string nombre_archivo, string modo [, int usar_ruta_inclusion])

Abre un archivo gzip (.gz) para lectura o escritura.

gzopen() puede ser usada para leer un archivo el cual no está en formato gzip; en este caso **gzread()** leerá directamente desde el archivo sin relizar descompresión.

Lista de parámetros

nombre_archivo

El nombre del archivo.

modo

Como en **fopen()** (*rb* o *wb*) pero puede también incluir un nivel de compresión (*wb9*) o una estrategia: *f* para datos filtrados como en *wb6f*, *h* para *Huffman only compression* como en *wb1h*. (Vea la descripción de deflateInit2 en `zlib.h` para más información acerca del parámetro de estrategia.)

usar_ruta_inclusion

Puede usar el tercer parámetro opcional y definirlo como "1", si desea que el archivo sea buscado también en include_path.

Valores retornados

Regresa un apuntador de archivo a el archivo abierto, después de eso, todo lo que se lea desde este descriptor de archivo será descomprimido transparentemente y lo que escriba será comprimido.

Si falla al abrir el archivo, la función regresa **`FALSE`**.

Ejemplos

Ejemplo 1. Ejemplo de gzopen()

```
<?php
$fp = gzopen("/tmp/file.gz", "r");
?>
```

Ver también

gzclose()

gzpassthru

(PHP 3, PHP 4 , PHP 5)

gzpassthru -- Imprimir todos los datos que restan en un apuntador a archivo gz

Descripción

int **gzpassthru** (resource zp)

Lee desde el apuntador de archivo gz hasta el final del archivo y escribe los resultados (sin compresión) a la salida estándar.

> **Nota:** Puede que necesite llamar **gzrewind()** para reiniciar el apuntador del archivo al principio del archivo si ya se ha escrito datos en él.
>
> **Sugerencia:** Si sólo desea volcar el contenido de un archivo al buffer de salida, sin modificarlo primero o buscar una posición en particular, puede que desee usar la función **readgzfile()**, la cual le ahorra la llamada a **gzopen()**.

Lista de parámetros

zp

> El apuntador de archivo debe ser válido, y debe apuntar a un archivo abierto satisfactoriamente por **gzopen()**.

Valores retornados

El número de caracteres sin compresión leídos desde *gz* y pasados a través de la entrada, o **`FALSE`** en caso de error.

Ejemplos

Ejemplo 1. Ejemplo de gzpassthru()

```
<?php
$fp = gzopen('file.gz', 'r');
gzpassthru($fp);
gzclose($fp);
?>
```

gzputs

gzputs -- Alias de **gzwrite()**

Descripción

Esta función es un alias de **gzwrite()**.

gzread

(PHP 3, PHP 4 , PHP 5)

gzread -- Lectura segura de archivo binario gz

Descripción

string **gzread** (resource zp, int longitud)

gzread() lee hasta *longitud* bytes de el apuntador de archivo gz. La lectura se detiene cuando *length* en bytes(sin compresión) han sido leídos o se encuentre el fin del archivo EOF, lo que ocurra primero.

Lista de parámetros

zp

El apuntador de archivo gz. Debe ser válido, y debe apuntar a un archivo exitosamente abierto por **gzopen()**.

longitud

El número de bytes a leer.

Valores retornados

Los datos que han sido leídos.

Ejemplos

Ejemplo 1. ejemplo de gzread()

```
<?php
// get contents of a gz-file into a string
$filename = "/usr/local/something.txt.gz";
$zd = gzopen($filename, "r");
$contents = gzread($zd, 10000);
gzclose($zd);
?>
```

Ver también

gzwrite()
gzopen()
gzgets()
gzgetss()
gzfile()
gzpassthru()

gzrewind

(PHP 3, PHP 4 , PHP 5)

gzrewind -- Reinicia la posición de un apuntador de archivo gz

Descripción

bool **gzrewind** (resource zp)

Establece el indicador de posición del apuntador de archivo vz al principio de el archivo.

Lista de parámetros

zp

El apuntador de archivo gz. Debe ser válido, y debe apuntar a un archivo exitosamente abierto por **gzopen()**.

Valores retornados

Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

Ver también

gzseek()
gztell()

gzseek

(PHP 3, PHP 4 , PHP 5)

gzseek -- Busca en el archivo gz apuntado

Descripción

int **gzseek** (resource zp, int offset)

Establece el indicador de posición del archivo para el apuntador de archivo. ñalente a ejecutar (en C) *gzseek(zp, offset, SEEK_SET)*.

Si el archivo es abierto para lectura, esta función es emulada pero puede ser extremadamente lenta. Si el archivo es abierto para escritura, solo se soportan bÃ‚Â´squedas hacia adelante; **gzseek()** entonces comprime una secuencia de ceros hasta la nueva posición.

Lista de parámetros

zp

El apuntador de archivo gz. Debe ser válido, y debe apuntar a un archivo exitosamente abierto por **gzopen()**.

offset

La posición a buscar.

Valores retornados

En caso de éxito, regresa 0; de otro modo, regresa -1. Note que si se pasa del fin del archivo no es considerado un error.

Ver también

gztell()
gzrewind()

gztell

(PHP 3, PHP 4 , PHP 5)

gztell -- Indica la posición de lectura/escritura del apuntador a archivo gz

Descripción

int **gztell** (resource zp)

Obtiene la posición del apuntador de archivo dado; ej., su desplazamiento secuencial en el interior del archivo.

Lista de parámetros

zp

El apuntador de archivo debe ser válido, y debe apuntar a un archivo abierto con éxito por **gzopen()**.

Valores retornados

La posición de el apuntador del rchivo o **FALSE** si ocurre un error.

Ver también

gzopen()
gzseek()
gzrewind()

gzuncompress

(PHP 4 >= 4.0.1, PHP 5)

gzuncompress -- Descomprime una cadena comprimida

Descripción

string **gzuncompress** (string datos [, int longitud])

Esta función descomprime una cadena comprimida.

Lista de parámetros

datos

Los datos comprimidos por **gzcompress()**.

longitud

La longitud máxima a decodificar o descomprimir.

Valores retornados

Los datos originales sin compresió o **`FALSE`** en error.

La función regresará un error si los datos sin comprimir son mas que 32768 veces la longitud de los *datos* de entrada, o mayor al parámetro opcional *longitud*.

Ver también

gzcompress()
gzinflate()
gzdeflate()
gzencode()

gzwrite

(PHP 3, PHP 4 , PHP 5)

gzwrite -- Escritura en un archivo gz, segura con material binario

Descripción

int **gzwrite** (resource zp, string cadena [, int longitud])

gzwrite() escribe el contenido de *cadena* al archivo gz dado.

Lista de parámetros

zp

El apuntador de archivo gz. Debe ser válido, y debe apuntar a un archivo abierto exitosamente por **gzopen()**.

cadena

La cadena a escribir.

longitud

El número de bytes sin comprimir a escribir. Si se define, la escritura se detendrá después de que se haya alcanzado la *longitud* de bytes escritos (sin comprimir) o que se encuentre el fin de *cadena*, lo que ocurra primero.

Nota: note que si el argumento *longitud* es dado, entoces la opción de configuración magic_quotes_runtime será ignorada no se retirarán los slashes ("/") de *cadena*.

Valores retornados

Regresa el número de bytes (sin comprimir) escritos a archivo gz.

Ver también

gzread()
gzopen()

readgzfile

(PHP 3, PHP 4 , PHP 5)

readgzfile -- Imprimir un archivo gz

Descripción

int **readgzfile** (string nombre_archivo [, int usar_ruta_inclusion])

Lee un archivo, lo descomprime y lo escribe en la salida estándar.

La función **readgzfile()** puede ser usada para leer un archivo que no se encuentra en formato gzip; en este caso **readgzfile()** leerá desde el archivo directamente sin descomprimirlo.

Lista de parámetros

nombre_archivo

El nombre del archivo. El archivo será abierto desde el sistema de archivos y su contenido será escrito en la salida estándar.

usar_ruta_inclusion

Puede usar el segundo parámetro opcional, y definirlo como *1*, si desea que el archivo sea buscado también en include_path.

Valores retornados

Devuelve el número de bytes (sin compresión) leídos desde el archivo. Si ocurre un error, se devuelve **FALSE** y, a menos que la función sea llamada como *@readgzfile*, se imprime un mensaje de error.

Ver también

gzpassthru()
gzfile()
gzopen()

zlib_get_coding_type

(PHP 4 >= 4.3.2, PHP 5)

zlib_get_coding_type -- Regresa el tipo de codificación usada para la salida de la compresión

Descripción

string **zlib_get_coding_type** (void)

Regresa el tipo de codificación usada en la compresión.

Valores retornados

Los posibles valores regresados son *gzip*, *deflate*, o **FALSE**.

Ver también

La directiva zlib.output_compression

VII. Zend API

Hacking the Core of PHP

> Those who know don't talk.
>
> Those who talk don't know.

Sometimes, PHP "as is" simply isn't enough. Although these cases are rare for the average user, professional applications will soon lead PHP to the edge of its capabilities, in terms of either speed or functionality. New functionality cannot always be implemented natively due to language restrictions and inconveniences that arise when having to carry around a huge library of default code appended to every single script, so another method needs to be found for overcoming these eventual lacks in PHP.

As soon as this point is reached, it's time to touch the heart of PHP and take a look at its core, the C code that makes PHP go.

Tabla de contenidos

Capítulo 44. Overview

"Extending PHP" is easier said than done. PHP has evolved to a full-fledged tool consisting of a few megabytes of source code, and to hack a system like this quite a few things have to be learned and considered. When structuring this chapter, we finally decided on the "learn by doing" approach. This is not the most scientific and professional approach, but the method that's the most fun and gives the best end results. In the following sections, you'll learn quickly how to get the most basic extensions to work almost instantly. After that, you'll learn about Zend's advanced API functionality. The alternative would have been to try to impart the functionality, design, tips, tricks, etc. as a whole, all at once, thus giving a complete look at the big picture before doing anything practical. Although this is the "better" method, as no dirty hacks have to be made, it can be very frustrating as well as energy- and time-consuming, which is why we've decided on the direct approach.

Note that even though this chapter tries to impart as much knowledge as possible about the inner workings of PHP, it's impossible to really give a complete guide to extending PHP that works 100% of the time in all cases. PHP is such a huge and complex package that its inner workings can only be understood if you make yourself familiar with it by practicing, so we encourage you to work with the source.

What Is Zend? and What Is PHP?

The name *Zend* refers to the language engine, PHP's core. The term *PHP* refers to the complete system as it appears from the outside. This might sound a bit confusing at first, but it's not that complicated (see Figura 44-1). To implement a Web script interpreter, you need three parts:

1. The *interpreter* part analyzes the input code, translates it, and executes it.

2. The *functionality* part implements the functionality of the language (its functions, etc.).

3. The *interface* part talks to the Web server, etc.

Zend takes part 1 completely and a bit of part 2; PHP takes parts 2 and 3. Together they form the complete PHP package. Zend itself really forms only the language core, implementing PHP at its very basics with some predefined functions. PHP contains all the modules that actually create the language's outstanding capabilities.

Figura 44-1. The internal structure of PHP.

The following sections discuss where PHP can be extended and how it's done.

Capítulo 45. Extension Possibilities

As shown in Figura 44-1 above, PHP can be extended primarily at three points: external modules, built-in modules, and the Zend engine. The following sections discuss these options.

External Modules

External modules can be loaded at script runtime using the function **dl()**. This function loads a shared object from disk and makes its functionality available to the script to which it's being bound. After the script is terminated, the external module is discarded from memory. This method has both advantages and disadvantages, as described in the following table:

Advantages	Disadvantages
External modules don't require recompiling of PHP.	The shared objects need to be loaded every time a script is being executed (every hit), which is very slow.
The size of PHP remains small by "outsourcing" certain functionality.	External additional files clutter up the disk.

	Every script that wants to use an external module's functionality has to specifically include a call to **dl()**, or the *extension* tag in `php.ini` needs to be modified (which is not always a suitable solution).

To sum up, external modules are great for third-party products, small additions to PHP that are rarely used, or just for testing purposes. To develop additional functionality quickly, external modules provide the best results. For frequent usage, larger implementations, and complex code, the disadvantages outweigh the advantages.

Third parties might consider using the *extension* tag in `php.ini` to create additional external modules to PHP. These external modules are completely detached from the main package, which is a very handy feature in commercial environments. Commercial distributors can simply ship disks or archives containing only their additional modules, without the need to create fixed and solid PHP binaries that don't allow other modules to be bound to them.

Built-in Modules

Built-in modules are compiled directly into PHP and carried around with every PHP process; their functionality is instantly available to every script that's being run. Like external modules, built-in modules have advantages and disadvantages, as described in the following table:

Advantages	Disadvantages
No need to load the module specifically; the functionality is instantly available.	Changes to built-in modules require recompiling of PHP.

No external files clutter up the disk; everything resides in the PHP binary.	The PHP binary grows and consumes more memory.

Built-in modules are best when you have a solid library of functions that remains relatively unchanged, requires better than poor-to-average performance, or is used frequently by many scripts on your site. The need to recompile PHP is quickly compensated by the benefit in speed and ease of use. However, built-in modules are not ideal when rapid development of small additions is required.

The Zend Engine

Of course, extensions can also be implemented directly in the Zend engine. This strategy is good if you need a change in the language behavior or require special functions to be built directly into the language core. In general, however, modifications to the Zend engine should be avoided. Changes here result in incompatibilities with the rest of the world, and hardly anyone will ever adapt to specially patched Zend engines. Modifications can't be detached from the main PHP sources and are overridden with the next update using the "official" source repositories. Therefore, this method is generally considered bad practice and, due to its rarity, is not covered in this book.

Capítulo 46. Source Layout

> **Nota:** Prior to working through the rest of this chapter, you should retrieve clean, unmodified source trees of your favorite Web server. We're working with Apache (available at http://www.apache.org/) and, of course, with PHP (available at http://www.php.net/ - does it need to be said?).
>
> Make sure that you can compile a working PHP environment by yourself! We won't go into this issue here, however, as you should already have this most basic ability when studying this chapter.

Before we start discussing code issues, you should familiarize yourself with the source tree to be able to quickly navigate through PHP's files. This is a must-have ability to implement and debug extensions.

The following table describes the contents of the major directories.

Directory	Contents

`php4`	Main PHP source files and main header files; here you'll find all of PHP's API definitions, macros, etc. (important). Everything else is below this directory.
`php4/ext`	Repository for dynamic and built-in modules; by default, these are the "official" PHP modules that have been integrated into the main source tree. From PHP 4.0, it's possible to compile these standard extensions as dynamic loadable modules (at least, those that support it).

`php4/main`	This directory contains the main php macros and definitions. (important)
`php4/pear`	Directory for the PHP Extension and Application Repository. This directory contains core PEAR files.
`php4/sapi`	Contains the code for the different server abstraction layers.
`php4/TSRM`	Location of the "Thread Safe Resource Manager" (TSRM) for Zend and PHP.

`php4/Zend`	Location of the Zend Engine files; here you'll find all of Zend's API definitions, macros, etc. (important).

Discussing all the files included in the PHP package is beyond the scope of this chapter. However, you should take a close look at the following files:

- `php4/main/php.h`, located in the main PHP directory. This file contains most of PHP's macro and API definitions.
- `php4/Zend/zend.h`, located in the main Zend directory. This file contains most of Zend's macros and definitions.
- `php4/Zend/zend_API.h`, also located in the Zend directory, which defines Zend's API.

You should also follow some sub-inclusions from these files; for example, the ones relating to the Zend executor, the PHP initialization file support, and such. After reading these files, take the time to navigate around the package a little to see the interdependencies of all files and modules - how they relate to each other and especially how they make use of each other. This also helps you to adapt to the coding style in which PHP is authored. To extend PHP, you should quickly adapt to this style.

Extension Conventions

Zend is built using certain conventions; to avoid breaking its standards, you should follow the rules described in the following sections.

Macros

For almost every important task, Zend ships predefined macros that are extremely handy. The tables and figures in the following sections describe most of the basic functions, structures, and macros. The macro definitions can be found mainly in `zend.h` and `zend_API.h`. We suggest that you take a close look at these files after having studied this chapter. (Although you can go ahead and read them now, not everything will make sense to you yet.)

Memory Management

Resource management is a crucial issue, especially in server software. One of the most valuable

resources is memory, and memory management should be handled with extreme care. Memory management has been partially abstracted in Zend, and you should stick to this abstraction for obvious reasons: Due to the abstraction, Zend gets full control over all memory allocations. Zend is able to determine whether a block is in use, automatically freeing unused blocks and blocks with lost references, and thus prevent memory leaks. The functions to be used are described in the following table:

Function	Description
emalloc()	Serves as replacement for **malloc()**.
efree()	Serves as replacement for **free()**.
estrdup()	Serves as replacement for **strdup()**.
estrndup()	Serves as replacement for **strndup()**. Faster than **estrdup()** and binary-safe. This is the recommended function to use if you know the string length prior to duplicating it.
ecalloc()	Serves as replacement for **calloc()**.
erealloc()	Serves as replacement for **realloc()**.

emalloc(), **estrdup()**, **estrndup()**, **ecalloc()**, and **erealloc()** allocate internal memory; **efree()** frees these previously allocated blocks. Memory handled by the **e*()** functions is considered local to the current process and is discarded as soon as the script executed by this process is terminated.

Aviso
To allocate resident memory that survives termination of the current script, you can use **malloc()** and **free()**. This should only be done with extreme care, however, and only in conjunction with demands of the Zend API; otherwise, you risk memory leaks.

Zend also features a thread-safe resource manager to provide better native support for multithreaded Web servers. This requires you to allocate local structures for all of your global variables to allow concurrent threads to be run. Because the thread-safe mode of Zend was not finished back when this was written, it is not yet extensively covered here.

Directory and File Functions

The following directory and file functions should be used in Zend modules. They behave exactly like their C counterparts, but provide virtual working directory support on the thread level.

Zend Function	Regular C Function
V_GETCWD()	**getcwd()**
V_FOPEN()	**fopen()**
V_OPEN()	**open()**
V_CHDIR()	**chdir()**
V_GETWD()	**getwd()**
V_CHDIR_FILE()	Takes a file path as an argument and changes the current working directory to that file's directory.
V_STAT()	**stat()**
V_LSTAT()	**lstat()**

String Handling

Strings are handled a bit differently by the Zend engine than other values such as integers, Booleans, etc., which don't require additional memory allocation for storing their values. If you want to return a string from a function, introduce a new string variable to the symbol table, or do something similar, you have to make sure that the memory the string will be occupying has previously been allocated, using the aforementioned **e*()** functions for allocation. (This might not make much sense to you yet; just keep it somewhere in your head for now - we'll get back to it shortly.)

Complex Types

Complex types such as arrays and objects require different treatment. Zend features a single API for these types - they're stored using hash tables.

> **Nota:** To reduce complexity in the following source examples, we're only working with simple types such as integers at first. A discussion about creating more advanced types follows later in this chapter.

Capítulo 47. PHP's Automatic Build System

PHP 4 features an automatic build system that's very flexible. All modules reside in a subdirectory of the `ext` directory. In addition to its own sources, each module consists of a config.m4 file, for extension configuration. (for example, see http://www.gnu.org/manual/m4/html_mono/m4.html)

All these stub files are generated automatically, along with `.cvsignore`, by a little shell script named `ext_skel` that resides in the `ext` directory. As argument it takes the name of the module that you want to create. The shell script then creates a directory of the same name, along with the appropriate stub files.

Step by step, the process looks like this:

```
:~/cvs/php4/ext:> ./ext_skel --extname=my_module
Creating directory my_module
Creating basic files: config.m4 .cvsignore my_module.c php_my_module.h CREDITS EXPERIME

To use your new extension, you will have to execute the following steps:

1.  $ cd ..
2.  $ vi ext/my_module/config.m4
3.  $ ./buildconf
4.  $ ./configure --[with|enable]-my_module
5.  $ make
6.  $ ./php -f ext/my_module/my_module.php
7.  $ vi ext/my_module/my_module.c
8.  $ make

Repeat steps 3-6 until you are satisfied with ext/my_module/config.m4 and
step 6 confirms that your module is compiled into PHP. Then, start writing
code and repeat the last two steps as often as necessary.
```

This instruction creates the aforementioned files. To include the new module in the automatic configuration and build process, you have to run `buildconf`, which regenerates the `configure` script by searching through the `ext` directory and including all found `config.m4` files.

The default `config.m4` shown in Ejemplo 47-1 is a bit more complex:

Ejemplo 47-1. The default `config.m4`.

```
dnl $Id: Extending_Zend_Build.xml,v 1.8 2002/10/10 18:13:11 imajes Exp $
dnl config.m4 for extension my_module

dnl Comments in this file start with the string 'dnl'.
dnl Remove where necessary. This file will not work
dnl without editing.

dnl If your extension references something external, use with:

dnl PHP_ARG_WITH(my_module, for my_module support,
dnl Make sure that the comment is aligned:
dnl [  --with-my_module             Include my_module support])

dnl Otherwise use enable:

dnl PHP_ARG_ENABLE(my_module, whether to enable my_module support,
dnl Make sure that the comment is aligned:
dnl [  --enable-my_module           Enable my_module support])

if test "$PHP_MY_MODULE" != "no"; then
  dnl Write more examples of tests here...

  dnl # --with-my_module -> check with-path
  dnl SEARCH_PATH="/usr/local /usr"     # you might want to change this
  dnl SEARCH_FOR="/include/my_module.h"  # you most likely want to change this
  dnl if test -r $PHP_MY_MODULE/; then # path given as parameter
  dnl   MY_MODULE_DIR=$PHP_MY_MODULE
  dnl else # search default path list
  dnl   AC_MSG_CHECKING([for my_module files in default path])
  dnl   for i in $SEARCH_PATH ; do
  dnl     if test -r $i/$SEARCH_FOR; then
  dnl       MY_MODULE_DIR=$i
  dnl       AC_MSG_RESULT(found in $i)
  dnl     fi
  dnl   done
  dnl fi
  dnl
  dnl if test -z "$MY_MODULE_DIR"; then
  dnl   AC_MSG_RESULT([not found])
  dnl   AC_MSG_ERROR([Please reinstall the my_module distribution])
  dnl fi

  dnl # --with-my_module -> add include path
  dnl PHP_ADD_INCLUDE($MY_MODULE_DIR/include)

  dnl # --with-my_module -> chech for lib and symbol presence
  dnl LIBNAME=my_module # you may want to change this
  dnl LIBSYMBOL=my_module # you most likely want to change this

  dnl PHP_CHECK_LIBRARY($LIBNAME,$LIBSYMBOL,
  dnl [
  dnl   PHP_ADD_LIBRARY_WITH_PATH($LIBNAME, $MY_MODULE_DIR/lib, MY_MODULE_SHARED_LIBADD
  dnl   AC_DEFINE(HAVE_MY_MODULELIB,1,[ ])
  dnl ],[
  dnl   AC_MSG_ERROR([wrong my_module lib version or lib not found])
  dnl ],[
  dnl   -L$MY_MODULE_DIR/lib -lm -ldl
  dnl ])
  dnl
  dnl PHP_SUBST(MY_MODULE_SHARED_LIBADD)

  PHP_NEW_EXTENSION(my_module, my_module.c, $ext_shared)
fi
```

If you're unfamiliar with M4 files (now is certainly a good time to get familiar), this might be a bit confusing at first; but it's actually quite easy.

Note: Everything prefixed with *dnl* is treated as a comment and is not parsed.

The `config.m4` file is responsible for parsing the command-line options passed to `configure` at configuration time. This means that it has to check for required external files and do similar configuration and setup tasks.

The default file creates two configuration directives in the `configure` script: *--with-my_module* and *--enable-my_module*. Use the first option when referring external files (such as the *--with-apache* directive that refers to the Apache directory). Use the second option when the user simply has to decide whether to enable your extension. Regardless of which option you use, you should uncomment the other, unnecessary one; that is, if you're using *--enable-my_module*, you should remove support for *--with-my_module*, and vice versa.

By default, the `config.m4` file created by `ext_skel` accepts both directives and automatically enables your extension. Enabling the extension is done by using the *PHP_EXTENSION* macro. To change the default behavior to include your module into the PHP binary when desired by the user (by explicitly specifying *--enable-my_module* or *--with-my_module*), change the test for *$PHP_MY_MODULE* to *== "yes"*:

```
if test "$PHP_MY_MODULE" == "yes"; then dnl
    Action.. PHP_EXTENSION(my_module, $ext_shared)
    fi
```

This would require you to use *--enable-my_module* each time when reconfiguring and recompiling PHP.

Note: Be sure to run `buildconf` every time you change `config.m4`!

We'll go into more details on the M4 macros available to your configuration scripts later in this chapter. For now, we'll simply use the default files.

Capítulo 48. Creating Extensions

We'll start with the creation of a very simple extension at first, which basically does nothing more than implement a function that returns the integer it receives as parameter. Ejemplo 48-1 shows the source.

Ejemplo 48-1. A simple extension.

```
/* include standard header */
#include "php.h"

/* declaration of functions to be exported */
ZEND_FUNCTION(first_module);

/* compiled function list so Zend knows what's in this module */
zend_function_entry firstmod_functions[] =
{
    ZEND_FE(first_module, NULL)
    {NULL, NULL, NULL}
};

/* compiled module information */
zend_module_entry firstmod_module_entry =
{
    STANDARD_MODULE_HEADER,
    "First Module",
    firstmod_functions,
    NULL,
    NULL,
    NULL,
    NULL,
    NULL,
    NO_VERSION_YET,
    STANDARD_MODULE_PROPERTIES
};

/* implement standard "stub" routine to introduce ourselves to Zend */
#if COMPILE_DL_FIRST_MODULE
ZEND_GET_MODULE(firstmod)
#endif

/* implement function that is meant to be made available to PHP */
ZEND_FUNCTION(first_module)
{
    long parameter;

    if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "l", &parameter) == FAILURE) {
        return;
    }

    RETURN_LONG(parameter);
}
```

This code contains a complete PHP module. We'll explain the source code in detail shortly, but first we'd like to discuss the build process. (This will allow the impatient to experiment before we dive into API discussions.)

> **Nota:** The example source makes use of some features introduced with the Zend version used in PHP 4.1.0 and above, it won't compile with older PHP 4.0.x versions.

Compiling Modules

There are basically two ways to compile modules:

- Use the provided "make" mechanism in the `ext` directory, which also allows building of dynamic loadable modules.
- Compile the sources manually.

The first method should definitely be favored, since, as of PHP 4.0, this has been standardized into a sophisticated build process. The fact that it is so sophisticated is also its drawback, unfortunately -

it's hard to understand at first. We'll provide a more detailed introduction to this later in the chapter, but first let's work with the default files.

The second method is good for those who (for some reason) don't have the full PHP source tree available, don't have access to all files, or just like to juggle with their keyboard. These cases should be extremely rare, but for the sake of completeness we'll also describe this method.

Compiling Using Make. To compile the sample sources using the standard mechanism, copy all their subdirectories to the `ext` directory of your PHP source tree. Then run `buildconf`, which will create an updated `configure` script containing appropriate options for the new extension. By default, all the sample sources are disabled, so you don't have to fear breaking your build process.

After you run `buildconf`, `configure --help` shows the following additional modules:

```
--enable-array_experiments   BOOK: Enables array experiments
  --enable-call_userland       BOOK: Enables userland module
  --enable-cross_conversion    BOOK: Enables cross-conversion module
  --enable-first_module        BOOK: Enables first module
  --enable-infoprint           BOOK: Enables infoprint module
  --enable-reference_test      BOOK: Enables reference test module
  --enable-resource_test       BOOK: Enables resource test module
  --enable-variable_creation   BOOK: Enables variable-creation module
```

The module shown earlier in Ejemplo 48-1 can be enabled with *--enable-first_module* or *--enable-first_module=yes*.

Compiling Manually. To compile your modules manually, you need the following commands:

Action	Command
Compiling	cc -fpic -DCOMPILE_DL=1 -I/usr/local/include -I. -I.. -I../Zend -c -o <your_object_file> <your_c_file>
Linking	cc -shared -L/usr/local/lib -rdynamic -o <your_module_file> <your_object_file(s)>

The command to compile the module simply instructs the compiler to generate position-independent code (*-fpic* shouldn't be omitted) and additionally defines the constant *COMPILE_DL* to tell the

module code that it's compiled as a dynamically loadable module (the test module above checks for this; we'll discuss it shortly). After these options, it specifies a number of standard include paths that should be used as the minimal set to compile the source files.

Note: All include paths in the example are relative to the directory `ext`. If you're compiling from another directory, change the pathnames accordingly. Required items are the PHP directory, the `Zend` directory, and (if necessary), the directory in which your module resides.

The link command is also a plain vanilla command instructing linkage as a dynamic module.

You can include optimization options in the compilation command, although these have been omitted in this example (but some are included in the makefile template described in an earlier section).

Note: Compiling and linking manually as a static module into the PHP binary involves very long instructions and thus is not discussed here. (It's not very efficient to type all those commands.)

Capítulo 49. Using Extensions

Depending on the build process you selected, you should either end up with a new PHP binary to be linked into your Web server (or run as CGI), or with an .so (shared object) file. If you compiled the example file `first_module.c` as a shared object, your result file should be `first_module.so`. To use it, you first have to copy it to a place from which it's accessible to PHP. For a simple test procedure, you can copy it to your `htdocs` directory and try it with the source in Ejemplo 49-1. If you compiled it into the PHP binary, omit the call to **dl()**, as the module's functionality is instantly available to your scripts.

Aviso
For security reasons, you *should not* put your dynamic modules into publicly accessible directories. Even though it *can* be done and it simplifies testing, you should put them into a separate directory in production environments.

Ejemplo 49-1. A test file for first_module.so.

```
<?php

// remove next comment if necessary
// dl("first_module.so");

$param = 2;
$return = first_module($param);

print("We sent '$param' and got '$return'");

?>
```

Calling this PHP file in your Web browser should give you the output shown in Figura 49-1.

Figura 49-1. Output of first_module.php.

If required, the dynamic loadable module is loaded by calling the **dl()** function. This function looks for the specified shared object, loads it, and makes its functions available to PHP. The module

exports the function **first_module()**, which accepts a single parameter, converts it to an integer, and returns the result of the conversion.

If you've gotten this far, congratulations! You just built your first extension to PHP.

Capítulo 50. Troubleshooting

Actually, not much troubleshooting can be done when compiling static or dynamic modules. The only problem that could arise is that the compiler will complain about missing definitions or something similar. In this case, make sure that all header files are available and that you specified their path correctly in the compilation command. To be sure that everything is located correctly, extract a clean PHP source tree and use the automatic build in the `ext` directory with the fresh files; this will guarantee a safe compilation environment. If this fails, try manual compilation.

PHP might also complain about missing functions in your module. (This shouldn't happen with the sample sources if you didn't modify them.) If the names of external functions you're trying to access from your module are misspelled, they'll remain as "unlinked symbols" in the symbol table. During dynamic loading and linkage by PHP, they won't resolve because of the typing errors - there are no corresponding symbols in the main binary. Look for incorrect declarations in your module file or incorrectly written external references. Note that this problem is specific to dynamic loadable modules; it doesn't occur with static modules. Errors in static modules show up at compile time.

Capítulo 51. Source Discussion

Now that you've got a safe build environment and you're able to include the modules into PHP files, it's time to discuss how everything works.

Module Structure

All PHP modules follow a common structure:

- Header file inclusions (to include all required macros, API definitions, etc.)
- C declaration of exported functions (required to declare the Zend function block)
- Declaration of the Zend function block
- Declaration of the Zend module block
- Implementation of **get_module()**
- Implementation of all exported functions

Header File Inclusions

The only header file you really have to include for your modules is `php.h`, located in the PHP directory. This file makes all macros and API definitions required to build new modules available to your code.

Tip: It's good practice to create a separate header file for your module that contains module-specific definitions. This header file should contain all the forward definitions for exported functions and include `php.h`. If you created your module using *ext_skel* you already have such a header file prepared.

Declaring Exported Functions

To declare functions that are to be exported (i.e., made available to PHP as new native functions), Zend provides a set of macros. A sample declaration looks like this:

```
ZEND_FUNCTION ( my_function );
```

ZEND_FUNCTION declares a new C function that complies with Zend's internal API. This means that the function is of type *void* and accepts *INTERNAL_FUNCTION_PARAMETERS* (another macro) as parameters. Additionally, it prefixes the function name with *zif*. The immediately expanded version of the above definitions would look like this:

```
void zif_my_function ( INTERNAL_FUNCTION_PARAMETERS );
```

Expanding *INTERNAL_FUNCTION_PARAMETERS* results in the following:

```
void zif_my_function( int ht
                    , zval * return_value
                    , zval * this_ptr
                    , int return_value_used
                    , zend_executor_globals * executor_globals
                    );
```

Since the interpreter and executor core have been separated from the main PHP package, a second API defining macros and function sets has evolved: the Zend API. As the Zend API now handles quite a few of the responsibilities that previously belonged to PHP, a lot of PHP functions have been reduced to macros aliasing to calls into the Zend API. The recommended practice is to use the Zend API wherever possible, as the old API is only preserved for compatibility reasons. For example, the types *zval* and *pval* are identical. *zval* is Zend's definition; *pval* is PHP's definition (actually, *pval* is an alias for *zval* now). As the macro *INTERNAL_FUNCTION_PARAMETERS* is a Zend macro, the above declaration contains *zval*. When writing code, you should always use *zval* to conform to the new Zend API.

The parameter list of this declaration is very important; you should keep these parameters in mind (see Tabla 51-1 for descriptions).

Tabla 51-1. Zend's Parameters to Functions Called from PHP

Parameter	Description

ht	The number of arguments passed to the Zend function. You should not touch this directly, but instead use ZEND_NUM_ARGS() to obtain the value.
return_value	This variable is used to pass any return values of your function back to PHP. Access to this variable is best done using the predefined macros. For a description of these see below.

this_ptr	Using this variable, you can gain access to the object in which your function is contained, if it's used within an object. Use the function **getThis()** to obtain this pointer.

return_value_used	This flag indicates whether an eventual return value from this function will actually be used by the calling script. *0* indicates that the return value is not used; *1* indicates that the caller expects a return value. Evaluation of this flag can be done to verify correct usage of the function as well as speed optimizations in case returning a value requires expensive operations (for an example, see how `array.c` makes use of this).

executor_globals	This variable points to global settings of the Zend engine. You'll find this useful when creating new variables, for example (more about this later). The executor globals can also be introduced to your function by using the macro *TSRMLS_FETCH()*.

Declaration of the Zend Function Block

Now that you have declared the functions to be exported, you also have to introduce them to Zend. Introducing the list of functions is done by using an array of *zend_function_entry*. This array consecutively contains all functions that are to be made available externally, with the function's name as it should appear in PHP and its name as defined in the C source. Internally, *zend_function_entry* is defined as shown in Ejemplo 51-1.

Ejemplo 51-1. Internal declaration of *zend_function_entry*.

```
typedef struct _zend_function_entry {
    char *fname;
    void (*handler)(INTERNAL_FUNCTION_PARAMETERS);
    unsigned char *func_arg_types;
} zend_function_entry;
```

Entry	Description
fname	Denotes the function name as seen in PHP (for example, *fopen*, *mysql_connect*, or, in our example, *first_module*).
handler	Pointer to the C function responsible for handling calls to this function. For example, see the standard macro *INTERNAL_FUNCTION_PARAMETERS* discussed earlier.
func_arg_types	Allows you to mark certain parameters so that they're forced to be passed by reference. You usually should set this to NULL.

In the example above, the declaration looks like this:

```
zend_function_entry firstmod_functions[] =
{
    ZEND_FE(first_module, NULL)
    {NULL, NULL, NULL}
};
```

You can see that the last entry in the list always has to be *{NULL, NULL, NULL}*. This marker has to be set for Zend to know when the end of the list of exported functions is reached.

> **Nota:** You *cannot* use the predefined macros for the end marker, as these would try to refer to a function named "NULL"!

The macro *ZEND_FE* (short for 'Zend Function Entry') simply expands to a structure entry in *zend_function_entry*. Note that these macros introduce a special naming scheme to your functions - your C functions will be prefixed with *zif_*, meaning that *ZEND_FE(first_module)* will refer to a C function **zif_first_module()**. If you want to mix macro usage with hand-coded entries (not a good practice), keep this in mind.

Tip: Compilation errors that refer to functions named **zif_*()** relate to functions defined with *ZEND_FE*.

Tabla 51-2 shows a list of all the macros that you can use to define functions.

Tabla 51-2. Macros for Defining Functions

Macro Name	Description
ZEND_FE(name, arg_types)	Defines a function entry of the name *name* in *zend_function_entry*. Requires a corresponding C function. *arg_types* needs to be set to *NULL*. This function uses automatic C function name generation by prefixing the PHP function name with *zif_*. For example, *ZEND_FE("first_module", NULL)* introduces a function **first_module()** to PHP and links it to the C function **zif_first_module()**. Use in conjunction with *ZEND_FUNCTION*.

ZEND_NAMED_FE (php_name, name, arg_types)	Defines a function that will be available to PHP by the name *php_name* and links it to the corresponding C function *name*. *arg_types* needs to be set to *NULL*. Use this function if you don't want the automatic name prefixing introduced by *ZEND_FE*. Use in conjunction with *ZEND_NAMED_FUNCTION*.
ZEND_FALIAS (name, alias, arg_types)	Defines an alias named *alias* for *name*. *arg_types* needs to be set to *NULL*. Doesn't require a corresponding C function; refers to the alias target instead.
PHP_FE (name, arg_types)	Old PHP API ñalent of *ZEND_FE*.

PHP_NAMED_FE (runtime_name, name, arg_types)	Old PHP API ñalent of *ZEND_NAMED_FE*.

Note: You can't use *ZEND_FE* in conjunction with *PHP_FUNCTION*, or *PHP_FE* in conjunction with *ZEND_FUNCTION*. However, it's perfectly legal to mix *ZEND_FE* and *ZEND_FUNCTION* with *PHP_FE* and *PHP_FUNCTION* when staying with the same macro set for each function to be declared. But mixing is *not* recommended; instead, you're advised to use the *ZEND_** macros only.

Declaration of the Zend Module Block

This block is stored in the structure *zend_module_entry* and contains all necessary information to describe the contents of this module to Zend. You can see the internal definition of this module in Ejemplo 51-2.

Ejemplo 51-2. Internal declaration of *zend_module_entry*.

```
typedef struct _zend_module_entry zend_module_entry;

    struct _zend_module_entry {
    unsigned short size;
    unsigned int zend_api;
    unsigned char zend_debug;
    unsigned char zts;
    char *name;
    zend_function_entry *functions;
    int (*module_startup_func)(INIT_FUNC_ARGS);
    int (*module_shutdown_func)(SHUTDOWN_FUNC_ARGS);
    int (*request_startup_func)(INIT_FUNC_ARGS);
    int (*request_shutdown_func)(SHUTDOWN_FUNC_ARGS);
    void (*info_func)(ZEND_MODULE_INFO_FUNC_ARGS);
    char *version;

[ Rest of the structure is not interesting here ]

};
```

Entry	Description
size, *zend_api*, *zend_debug* and *zts*	Usually filled with the *"STANDARD_MODULE_HEADER"*, which fills these four members with the size of the whole zend_module_entry, the *ZEND_MODULE_API_NO*, whether it is a debug build or normal build (*ZEND_DEBUG*) and if ZTS is enabled (*USING_ZTS*).
name	Contains the module name (for example, *"File functions"*, *"Socket functions"*, *"Crypt"*, etc.). This name will show up in **phpinfo()**, in the section "Additional Modules."
functions	Points to the Zend function block, discussed in the preceding section.
module_startup_func	This function is called once upon module initialization and can be used to do one-time initialization steps (such as initial memory allocation, etc.). To indicate a failure during initialization, return *FAILURE*; otherwise,

In our example, this structure is implemented as follows:

```
zend_module_entry firstmod_module_entry =
{
    STANDARD_MODULE_HEADER,
    "First Module",
    firstmod_functions,
    NULL, NULL, NULL, NULL, NULL,
    NO_VERSION_YET,
    STANDARD_MODULE_PROPERTIES,
};
```

This is basically the easiest and most minimal set of values you could ever use. The module name is set to *First Module*, then the function list is referenced, after which all startup and shutdown functions are marked as being unused.

For reference purposes, you can find a list of the macros involved in declared startup and shutdown functions in Tabla 51-3. These are not used in our basic example yet, but will be demonstrated later on. You should make use of these macros to declare your startup and shutdown functions, as these require special arguments to be passed (*INIT_FUNC_ARGS* and *SHUTDOWN_FUNC_ARGS*), which are automatically included into the function declaration when using the predefined macros. If you declare your functions manually and the PHP developers decide that a change in the argument list is necessary, you'll have to change your module sources to remain compatible.

Tabla 51-3. Macros to Declare Startup and Shutdown Functions

Macro	Description
ZEND_MINIT (module)	Declares a function for module startup. The generated name will be *zend_minit_<module>* (for example, *zend_minit_first_module*). Use in conjunction with *ZEND_MINIT_FUNCTION*.
ZEND_MSHUTDOWN(module)	Declares a function for module shutdown. The generated name will be *zend_mshutdown_<module>* (for example, *zend_mshutdown_first_module*). Use in conjunction with *ZEND_MSHUTDOWN_FUNCTION*.

ZEND_RINIT (module)	Declares a function for request startup. The generated name will be *zend_rinit_<module>* (for example, *zend_rinit_first_module*). Use in conjunction with *ZEND_RINIT_FUNCTION*.
ZEND_RSHUTDOWN(module)	Declares a function for request shutdown. The generated name will be *zend_rshutdown_<module>* (for example, *zend_rshutdown_first_module*). Use in conjunction with *ZEND_RSHUTDOWN_FUNCTION*.
ZEND_MINFO (module)	Declares a function for printing module information, used when **phpinfo()** is called. The generated name will be *zend_info_<module>* (for example, *zend_info_first_module*). Use in conjunction with *ZEND_MINFO_FUNCTION*.

Creation of get_module()

This function is special to all dynamic loadable modules. Take a look at the creation via the *ZEND_GET_MODULE* macro first:

```
#if COMPILE_DL_FIRSTMOD
     ZEND_GET_MODULE(firstmod)
#endif
```

The function implementation is surrounded by a conditional compilation statement. This is needed since the function **get_module()** is only required if your module is built as a dynamic extension. By specifying a definition of *COMPILE_DL_FIRSTMOD* in the compiler command (see above for a discussion of the compilation instructions required to build a dynamic extension), you can instruct your module whether you intend to build it as a dynamic extension or as a built-in module. If you want a built-in module, the implementation of **get_module()** is simply left out.

get_module() is called by Zend at load time of the module. You can think of it as being invoked by the **dl()** call in your script. Its purpose is to pass the module information block back to Zend in order to inform the engine about the module contents.

If you don't implement a **get_module()** function in your dynamic loadable module, Zend will compliment you with an error message when trying to access it.

Implementation of All Exported Functions

Implementing the exported functions is the final step. The example function in *first_module* looks like this:

```
ZEND_FUNCTION(first_module)
{
    long parameter;

    if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "l", &parameter) == FAILURE) {
        return;
    }

    RETURN_LONG(parameter);
}
```

The function declaration is done using *ZEND_FUNCTION*, which corresponds to *ZEND_FE* in the function entry table (discussed earlier).

After the declaration, code for checking and retrieving the function's arguments, argument conversion, and return value generation follows (more on this later).

Summary

That's it, basically - there's nothing more to implementing PHP modules. Built-in modules are structured similarly to dynamic modules, so, equipped with the information presented in the previous sections, you'll be able to fight the odds when encountering PHP module source files.

Now, in the following sections, read on about how to make use of PHP's internals to build powerful extensions.

Capítulo 52. Accepting Arguments

One of the most important issues for language extensions is accepting and dealing with data passed

via arguments. Most extensions are built to deal with specific input data (or require parameters to perform their specific actions), and function arguments are the only real way to exchange data between the PHP level and the C level. Of course, there's also the possibility of exchanging data using predefined global values (which is also discussed later), but this should be avoided by all means, as it's extremely bad practice.

PHP doesn't make use of any formal function declarations; this is why call syntax is always completely dynamic and never checked for errors. Checking for correct call syntax is left to the user code. For example, it's possible to call a function using only one argument at one time and four arguments the next time - both invocations are syntactically absolutely correct.

Determining the Number of Arguments

Since PHP doesn't have formal function definitions with support for call syntax checking, and since PHP features variable arguments, sometimes you need to find out with how many arguments your function has been called. You can use the *ZEND_NUM_ARGS* macro in this case. In previous versions of PHP, this macro retrieved the number of arguments with which the function has been called based on the function's hash table entry, *ht*, which is passed in the *INTERNAL_FUNCTION_PARAMETERS* list. As *ht* itself now contains the number of arguments that have been passed to the function, *ZEND_NUM_ARGS* has been stripped down to a dummy macro (see its definition in `zend_API.h`). But it's still good practice to use it, to remain compatible with future changes in the call interface. *Note:* The old PHP ñalent of this macro is *ARG_COUNT*.

The following code checks for the correct number of arguments:

```
if(ZEND_NUM_ARGS() != 2) WRONG_PARAM_COUNT;
```

If the function is not called with two arguments, it exits with an error message. The code snippet above makes use of the tool macro *WRONG_PARAM_COUNT*, which can be used to generate a standard error message (see Figura 52-1).

Figura 52-1. *WRONG_PARAM_COUNT* in action.

This macro prints a default error message and then returns to the caller. Its definition can also be found in `zend_API.h` and looks like this:

```
ZEND_API void wrong_param_count(void);

#define WRONG_PARAM_COUNT { wrong_param_count(); return; }
```

As you can see, it calls an internal function named **wrong_param_count()** that's responsible for printing the warning. For details on generating customized error messages, see the later section "Printing Information."

Retrieving Arguments

> **New parameter parsing API:** This chapter documents the new Zend parameter parsing API introduced by Andrei Zmievski. It was introduced in the development stage between PHP 4.0.6 and 4.1.0 .

Parsing parameters is a very common operation and it may get a bit tedious. It would also be nice to have standardized error checking and error messages. Since PHP 4.1.0, there is a way to do just that by using the new parameter parsing API. It greatly simplifies the process of receiving parameteres, but it has a drawback in that it can't be used for functions that expect variable number of parameters. But since the vast majority of functions do not fall into those categories, this parsing API is recommended as the new standard way.

The prototype for parameter parsing function looks like this:

```
int zend_parse_parameters(int num_args TSRMLS_DC, char *type_spec, ...);
```

The first argument to this function is supposed to be the number of actual parameters passed to your function, so *ZEND_NUM_ARGS()* can be used for that. The second parameter should always be *TSRMLS_CC* macro. The third argument is a string that specifies the number and types of arguments your function is expecting, similar to how printf format string specifies the number and format of the output values it should operate on. And finally the rest of the arguments are pointers to variables which should receive the values from the parameters.

zend_parse_parameters() also performs type conversions whenever possible, so that you always receive the data in the format you asked for. Any type of scalar can be converted to another one, but conversions between complex types (arrays, objects, and resources) and scalar types are not allowed.

If the parameters could be obtained successfully and there were no errors during type conversion, the function will return *SUCCESS*, otherwise it will return *FAILURE*. The function will output informative error messages, if the number of received parameters does not match the requested number, or if type conversion could not be performed.

Here are some sample error messages:

```
Warning - ini_get_all() requires at most 1 parameter, 2 given
     Warning - wddx_deserialize() expects parameter 1 to be string, array given
```

Of course each error message is accompanied by the filename and line number on which it occurs.

Here is the full list of type specifiers:

- *l* - long
- *d* - double
- *s* - string (with possible null bytes) and its length
- *b* - boolean
- *r* - resource, stored in *zval**
- *a* - array, stored in *zval**
- *o* - object (of any class), stored in *zval**
- *O* - object (of class specified by class entry), stored in *zval**
- *z* - the actual *zval**

The following characters also have a meaning in the specifier string:

- | - indicates that the remaining parameters are optional. The storage variables corresponding

to these parameters should be initialized to default values by the extension, since they will not be touched by the parsing function if the parameters are not passed.

- */* - the parsing function will call **SEPARATE_ZVAL_IF_NOT_REF()** on the parameter it follows, to provide a copy of the parameter, unless it's a reference.

- *!* - the parameter it follows can be of specified type or *NULL* (only applies to a, o, O, r, and z). If *NULL* value is passed by the user, the storage pointer will be set to *NULL*.

The best way to illustrate the usage of this function is through examples:

```
/* Gets a long, a string and its length, and a zval. */
long l;
char *s;
int s_len;
zval *param;
if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC,
                          "lsz", &l, &s, &s_len, &param) == FAILURE) {
    return;
}

/* Gets an object of class specified by my_ce, and an optional double. */
zval *obj;
double d = 0.5;
if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC,
                          "O|d", &obj, my_ce, &d) == FAILURE) {
    return;
}

/* Gets an object or null, and an array.
   If null is passed for object, obj will be set to NULL. */
zval *obj;
zval *arr;
if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "O!a", &obj, &arr) == FAILURE) {
    return;
}

/* Gets a separated array. */
zval *arr;
if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "a/", &arr) == FAILURE) {
    return;
}

/* Get only the first three parameters (useful for varargs functions). */
zval *z;
zend_bool b;
zval *r;
if (zend_parse_parameters(3, "zbr!", &z, &b, &r) == FAILURE) {
    return;
}
```

Note that in the last example we pass 3 for the number of received parameters, instead of **ZEND_NUM_ARGS()**. What this lets us do is receive the least number of parameters if our function expects a variable number of them. Of course, if you want to operate on the rest of the parameters, you will have to use **zend_get_parameters_array_ex()** to obtain them.

The parsing function has an extended version that allows for an additional flags argument that controls its actions.

```
int zend_parse_parameters_ex(int flags, int num_args TSRMLS_DC, char *type_spec, ...);
```

The only flag you can pass currently is *ZEND_PARSE_PARAMS_QUIET*, which instructs the function to not output any error messages during its operation. This is useful for functions that expect several sets of completely different arguments, but you will have to output your own error messages.

For example, here is how you would get either a set of three longs or a string:

```
long l1, l2, l3;
char *s;
if (zend_parse_parameters_ex(ZEND_PARSE_PARAMS_QUIET,
                             ZEND_NUM_ARGS() TSRMLS_CC,
                             "lll", &l1, &l2, &l3) == SUCCESS) {
    /* manipulate longs */
} else if (zend_parse_parameters_ex(ZEND_PARSE_PARAMS_QUIET,
                                    ZEND_NUM_ARGS(), "s", &s, &s_len) == SUCCESS) {
    /* manipulate string */
} else {
    php_error(E_WARNING, "%s() takes either three long values or a string as argument",
              get_active_function_name(TSRMLS_C));
    return;
}
```

With all the abovementioned ways of receiving function parameters you should have a good handle on this process. For even more example, look through the source code for extensions that are shipped with PHP - they illustrate every conceivable situation.

Old way of retrieving arguments (deprecated)

> **Deprecated parameter parsing API:** This API is deprecated and superseded by the new ZEND parameter parsing API.

After having checked the number of arguments, you need to get access to the arguments themselves. This is done with the help of **zend_get_parameters_ex()**:

```
zval **parameter;

if(zend_get_parameters_ex(1, &parameter) != SUCCESS)
  WRONG_PARAM_COUNT;
```

All arguments are stored in a *zval* container, which needs to be pointed to *twice*. The snippet above tries to retrieve one argument and make it available to us via the *parameter* pointer.

zend_get_parameters_ex() accepts at least two arguments. The first argument is the number of arguments to retrieve (which should match the number of arguments with which the function has been called; this is why it's important to check for correct call syntax). The second argument (and all following arguments) are pointers to pointers to pointers to *zval*s. (Confusing, isn't it?) All these pointers are required because Zend works internally with ***zval*; to adjust a local ***zval* in our function,**zend_get_parameters_ex()** requires a pointer to it.

The return value of **zend_get_parameters_ex()** can either be *SUCCESS* or *FAILURE*, indicating (unsurprisingly) success or failure of the argument processing. A failure is most likely related to an incorrect number of arguments being specified, in which case you should exit with *WRONG_PARAM_COUNT*.

To retrieve more than one argument, you can use a similar snippet:

```
zval **param1, **param2, **param3, **param4;

if(zend_get_parameters_ex(4, &param1, &param2, &param3, &param4) != SUCCESS)
    WRONG_PARAM_COUNT;
```

zend_get_parameters_ex() only checks whether you're trying to retrieve too many parameters. If the function is called with five arguments, but you're only retrieving three of them with **zend_get_parameters_ex()**, you won't get an error but will get the first three parameters instead. Subsequent calls of **zend_get_parameters_ex()** won't retrieve the remaining arguments, but will get the same arguments again.

Dealing with a Variable Number of Arguments/Optional Parameters

If your function is meant to accept a variable number of arguments, the snippets just described are sometimes suboptimal solutions. You have to create a line calling **zend_get_parameters_ex()** for every possible number of arguments, which is often unsatisfying.

For this case, you can use the function **zend_get_parameters_array_ex()**, which accepts the number of parameters to retrieve and an array in which to store them:

```
zval **parameter_array[4];

/* get the number of arguments */
argument_count = ZEND_NUM_ARGS();

/* see if it satisfies our minimal request (2 arguments) */
/* and our maximal acceptance (4 arguments) */
if(argument_count < 2 || argument_count > 5)
    WRONG_PARAM_COUNT;

/* argument count is correct, now retrieve arguments */
if(zend_get_parameters_array_ex(argument_count, parameter_array) != SUCCESS)
    WRONG_PARAM_COUNT;
```

First, the number of arguments is checked to make sure that it's in the accepted range. After that, **zend_get_parameters_array_ex()** is used to fill *parameter_array* with valid pointers to the argument values.

A very clever implementation of this can be found in the code handling PHP's **fsockopen()** located in `ext/standard/fsock.c`, as shown in Ejemplo 52-1. Don't worry if you don't know all the functions used in this source yet; we'll get to them shortly.

Ejemplo 52-1. PHP's implementation of variable arguments in fsockopen().

```
pval **args[5];
int *sock=emalloc(sizeof(int));
int *sockp;
int arg_count=ARG_COUNT(ht);
int socketd = -1;
unsigned char udp = 0;
struct timeval timeout = { 60, 0 };
unsigned short portno;
unsigned long conv;
char *key = NULL;
FLS_FETCH();

if (arg_count > 5 || arg_count < 2 || zend_get_parameters_array_ex(arg_count,args)==FAI
    CLOSE_SOCK(1);
    WRONG_PARAM_COUNT;
}

switch(arg_count) {
    case 5:
        convert_to_double_ex(args[4]);
        conv = (unsigned long) (Z_DVAL_P(args[4]) * 1000000.0);
        timeout.tv_sec = conv / 1000000;
        timeout.tv_usec = conv % 1000000;
        /* fall-through */
    case 4:
        if (!PZVAL_IS_REF(*args[3])) {
            php_error(E_WARNING,"error string argument to fsockopen not passed by refer
        }
        pval_copy_constructor(*args[3]);
        ZVAL_EMPTY_STRING(*args[3]);
        /* fall-through */
    case 3:
        if (!PZVAL_IS_REF(*args[2])) {
            php_error(E_WARNING,"error argument to fsockopen not passed by reference");
            return;
        }
        ZVAL_LONG(*args[2], 0);
        break;
}

convert_to_string_ex(args[0]);
convert_to_long_ex(args[1]);
portno = (unsigned short) Z_LVAL_P(args[1]);

key = emalloc(Z_STRLEN_P(args[0]) + 10);
```

fsockopen() accepts two, three, four, or five parameters. After the obligatory variable declarations, the function checks for the correct range of arguments. Then it uses a fall-through mechanism in a *switch()* statement to deal with all arguments. The *switch()* statement starts with the maximum number of arguments being passed (five). After that, it automatically processes the case of four arguments being passed, then three, by omitting the otherwise obligatory *break* keyword in all stages. After having processed the last case, it exits the *switch()* statement and does the minimal argument processing needed if the function is invoked with only two arguments.

This multiple-stage type of processing, similar to a stairway, allows convenient processing of a variable number of arguments.

Accessing Arguments

To access arguments, it's necessary for each argument to have a clearly defined type. Again, PHP's extremely dynamic nature introduces some quirks. Because PHP never does any kind of type checking, it's possible for a caller to pass any kind of data to your functions, whether you want it or not. If you expect an integer, for example, the caller might pass an array, and vice versa - PHP

simply won't notice.

To work around this, you have to use a set of API functions to force a type conversion on every argument that's being passed (see Tabla 52-1).

Note: All conversion functions expect a ***zval* as parameter.

Tabla 52-1. Argument Conversion Functions

Function	Description
convert_to_boolean_ex()	Forces conversion to a Boolean type. Boolean values remain untouched. Longs, doubles, and strings containing *0* as well as NULL values will result in Boolean *0* (FALSE). Arrays and objects are converted based on the number of entries or properties, respectively, that they have. Empty arrays and objects are converted to FALSE; otherwise, to TRUE. All other values result in a Boolean *1* (TRUE).

convert_to_long_ex()	Forces conversion to a long, the default integer type. NULL values, Booleans, resources, and of course longs remain untouched. Doubles are truncated. Strings containing an integer are converted to their corresponding numeric representation, otherwise resulting in *0*. Arrays and objects are converted to *0* if empty, *1* otherwise.

convert_to_double_ex()	Forces conversion to a double, the default floating-point type. NULL values, Booleans, resources, longs, and of course doubles remain untouched. Strings containing a number are converted to their corresponding numeric representation, otherwise resulting in *0.0*. Arrays and objects are converted to *0.0* if empty, *1.0* otherwise.

convert_to_string_ex()	Forces conversion to a string. Strings remain untouched. NULL values are converted to an empty string. Booleans containing TRUE are converted to *"1"*, otherwise resulting in an empty string. Longs and doubles are converted to their corresponding string representation. Arrays are converted to the string *"Array"* and objects to the string *"Object"*.

convert_to_array_ex(value)	Forces conversion to an array. Arrays remain untouched. Objects are converted to an array by assigning all their properties to the array table. All property names are used as keys, property contents as values. NULL values are converted to an empty array. All other values are converted to an array that contains the specific source value in the element with the key *0*.

convert_to_object_ex (value)	Forces conversion to an object. Objects remain untouched. NULL values are converted to an empty object. Arrays are converted to objects by introducing their keys as properties into the objects and their values as corresponding property contents in the object. All other types result in an object with the property *scalar* , having the corresponding source value as content.
convert_to_null_ex(value)	Forces the type to become a NULL value, meaning empty.

Nota: You can find a demonstration of the behavior in `cross_conversion.php` on the accompanying CD-ROM. Figura 52-2 shows the output.

Figura 52-2. Cross-conversion behavior of PHP.

Using these functions on your arguments will ensure type safety for all data that's passed to you. If the supplied type doesn't match the required type, PHP forces dummy contents on the resulting value (empty strings, arrays, or objects, *0* for numeric values, *FALSE* for Booleans) to ensure a

defined state.

Following is a quote from the sample module discussed previously, which makes use of the conversion functions:

```
zval **parameter;

if((ZEND_NUM_ARGS() != 1) || (zend_get_parameters_ex(1, &parameter) != SUCCESS))
{
    WRONG_PARAM_COUNT;
}

convert_to_long_ex(parameter);

RETURN_LONG(Z_LVAL_P(parameter));
```

After retrieving the parameter pointer, the parameter value is converted to a long (an integer), which also forms the return value of this function. Understanding access to the contents of the value requires a short discussion of the *zval* type, whose definition is shown in Ejemplo 52-2.

Ejemplo 52-2. PHP/Zend *zval* type definition.

```
typedef pval zval;

typedef struct _zval_struct zval;

typedef union _zvalue_value {
        long lval;                                      /* long value */
        double dval;                                    /* double value */
        struct {
                char *val;
                int len;
        } str;
        HashTable *ht;                                  /* hash table value */
        struct {
                zend_class_entry *ce;
                HashTable *properties;
        } obj;
} zvalue_value;

struct _zval_struct {
        /* Variable information */
        zvalue_value value;             /* value */
        unsigned char type;     /* active type */
        unsigned char is_ref;
        short refcount;
};
```

Actually, *pval* (defined in `php.h`) is only an alias of *zval* (defined in `zend.h`), which in turn refers to *_zval_struct*. This is a most interesting structure. *_zval_struct* is the "master" structure, containing the value structure, type, and reference information. The substructure *zvalue_value* is a union that contains the variable's contents. Depending on the variable's type, you'll have to access different members of this union. For a description of both structures, see Tabla 52-2, Tabla 52-3 and Tabla 52-4.

Tabla 52-2. Zend *zval* Structure

Entry	Description

value	Union containing this variable's contents. See Tabla 52-3 for a description.
type	Contains this variable's type. For a list of available types, see Tabla 52-4.
is_ref	0 means that this variable is not a reference; 1 means that this variable is a reference to another variable.

refcount	The number of references that exist for this variable. For every new reference to the value stored in this variable, this counter is increased by 1. For every lost reference, this counter is decreased by 1. When the reference counter reaches 0, no references exist to this value anymore, which causes automatic freeing of the value.

Tabla 52-3. Zend *zvalue_value* Structure

Entry	Description
lval	Use this property if the variable is of the type *IS_LONG*, *IS_BOOLEAN*, or *IS_RESOURCE*.

dval	Use this property if the variable is of the type *IS_DOUBLE*.
str	This structure can be used to access variables of the type *IS_STRING*. The member *len* contains the string length; the member *val* points to the string itself. Zend uses C strings; thus, the string length contains a trailing *0x00*.
ht	This entry points to the variable's hash table entry if the variable is an array.
obj	Use this property if the variable is of the type *IS_OBJECT*.

Tabla 52-4. Zend Variable Type Constants

Constant	Description
IS_NULL	Denotes a NULL (empty) value.
IS_LONG	A long (integer) value.
IS_DOUBLE	A double (floating point) value.
IS_STRING	A string.

IS_ARRAY	Denotes an array.
IS_OBJECT	An object.
IS_BOOL	A Boolean value.
IS_RESOURCE	A resource (for a discussion of resources, see the appropriate section below).
IS_CONSTANT	A constant (defined) value.

To access a long you access *zval.value.lval*, to access a double you use *zval.value.dval*, and so on. Because all values are stored in a union, trying to access data with incorrect union members results in meaningless output.

Accessing arrays and objects is a bit more complicated and is discussed later.

Dealing with Arguments Passed by Reference

If your function accepts arguments passed by reference that you intend to modify, you need to take some precautions.

What we didn't say yet is that under the circumstances presented so far, you don't have write access to any *zval* containers designating function parameters that have been passed to you. Of course, you can change any *zval* containers that you created within your function, but you mustn't change any *zval*s that refer to Zend-internal data!

We've only discussed the so-called ***_ex()** API so far. You may have noticed that the API functions we've used are called **zend_get_parameters_ex()** instead of **zend_get_parameters()**, **convert_to_long_ex()** instead of **convert_to_long()**, etc. The ***_ex()** functions form the so-called new "extended" Zend API. They give a minor speed increase over the old API, but as a tradeoff are only meant for providing read-only access.

Because Zend works internally with references, different variables may reference the same value. Write access to a *zval* container requires this container to contain an isolated value, meaning a value that's not referenced by any other containers. If a *zval* container were referenced by other containers and you changed the referenced *zval*, you would automatically change the contents of the other containers referencing this *zval* (because they'd simply point to the changed value and thus change their own value as well).

zend_get_parameters_ex() doesn't care about this situation, but simply returns a pointer to the desired *zval* containers, whether they consist of references or not. Its corresponding function in the traditional API, **zend_get_parameters()**, immediately checks for referenced values. If it finds a reference, it creates a new, isolated *zval* container; copies the referenced data into this newly

allocated space; and then returns a pointer to the new, isolated value.

This action is called *zval separation* (or pval separation). Because the ***_ex()** API doesn't perform zval separation, it's considerably faster, while at the same time disabling write access.

To change parameters, however, write access is required. Zend deals with this situation in a special way: Whenever a parameter to a function is passed by reference, it performs automatic zval separation. This means that whenever you're calling a function like this in PHP, Zend will automatically ensure that *$parameter* is being passed as an isolated value, rendering it to a write-safe state:

```
my_function(&$parameter);
```

But this *is not* the case with regular parameters! All other parameters that are not passed by reference are in a read-only state.

This requires you to make sure that you're really working with a reference - otherwise you might produce unwanted results. To check for a parameter being passed by reference, you can use the macro *PZVAL_IS_REF*. This macro accepts a *zval** to check if it is a reference or not. Examples are given in in Ejemplo 52-3.

Ejemplo 52-3. Testing for referenced parameter passing.

```
zval *parameter;

if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "z", &parameter) == FAILURE)
    return;

/* check for parameter being passed by reference */
if (!PZVAL_IS_REF(*parameter)) {
{
    zend_error(E_WARNING, "Parameter wasn't passed by reference");
    RETURN_NULL();
}

/* make changes to the parameter */
ZVAL_LONG(*parameter, 10);
```

Assuring Write Safety for Other Parameters

You might run into a situation in which you need write access to a parameter that's retrieved with **zend_get_parameters_ex()** but not passed by reference. For this case, you can use the macro *SEPARATE_ZVAL*, which does a zval separation on the provided container. The newly generated *zval* is detached from internal data and has only a local scope, meaning that it can be changed or destroyed without implying global changes in the script context:

```
zval **parameter;

/* retrieve parameter */
zend_get_parameters_ex(1, &parameter);

/* at this stage, <parameter> still is connected */
/* to Zend's internal data buffers */

/* make <parameter> write-safe */
SEPARATE_ZVAL(parameter);

/* now we can safely modify <parameter> */
/* without implying global changes */
```

SEPARATE_ZVAL uses **emalloc()** to allocate the new *zval* container, which means that even if you

don't deallocate this memory yourself, it will be destroyed automatically upon script termination. However, doing a lot of calls to this macro without freeing the resulting containers will clutter up your RAM.

Note: As you can easily work around the lack of write access in the "traditional" API (with **zend_get_parameters()** and so on), this API seems to be obsolete, and is not discussed further in this chapter.

Capítulo 53. Creating Variables

When exchanging data from your own extensions with PHP scripts, one of the most important issues is the creation of variables. This section shows you how to deal with the variable types that PHP supports.

Overview

To create new variables that can be seen "from the outside" by the executing script, you need to allocate a new *zval* container, fill this container with meaningful values, and then introduce it to Zend's internal symbol table. This basic process is common to all variable creations:

```
zval *new_variable;

/* allocate and initialize new container */
MAKE_STD_ZVAL(new_variable);

/* set type and variable contents here, see the following sections */

/* introduce this variable by the name "new_variable_name" into the symbol table */
ZEND_SET_SYMBOL(EG(active_symbol_table), "new_variable_name", new_variable);

/* the variable is now accessible to the script by using $new_variable_name */
```

The macro *MAKE_STD_ZVAL* allocates a new *zval* container using *ALLOC_ZVAL* and initializes it using *INIT_ZVAL*. As implemented in Zend at the time of this writing, *initializing* means setting the reference count to *1* and clearing the *is_ref* flag, but this process could be extended later - this is why it's a good idea to keep using *MAKE_STD_ZVAL* instead of only using *ALLOC_ZVAL*. If you want to optimize for speed (and you don't have to explicitly initialize the *zval* container here), you can use *ALLOC_ZVAL*, but this isn't recommended because it doesn't ensure data integrity.

ZEND_SET_SYMBOL takes care of introducing the new variable to Zend's symbol table. This macro checks whether the value already exists in the symbol table and converts the new symbol to a reference if so (with automatic deallocation of the old *zval* container). This is the preferred method if speed is not a crucial issue and you'd like to keep memory usage low.

Note that *ZEND_SET_SYMBOL* makes use of the Zend executor globals via the macro *EG*. By specifying *EG(active_symbol_table)*, you get access to the currently active symbol table, dealing with the active, local scope. The local scope may differ depending on whether the function was invoked from within a function.

If you need to optimize for speed and don't care about optimal memory usage, you can omit the check for an existing variable with the same value and instead force insertion into the symbol table by using **zend_hash_update()**:

```
zval *new_variable;

/* allocate and initialize new container */
MAKE_STD_ZVAL(new_variable);

/* set type and variable contents here, see the following sections */

/* introduce this variable by the name "new_variable_name" into the symbol table */
zend_hash_update(
    EG(active_symbol_table),
    "new_variable_name",
    strlen("new_variable_name") + 1,
    &new_variable,
    sizeof(zval *),
    NULL
);
```

This is actually the standard method used in most modules.

The variables generated with the snippet above will always be of local scope, so they reside in the context in which the function has been called. To create new variables in the global scope, use the same method but refer to another symbol table:

```
zval *new_variable;

// allocate and initialize new container
MAKE_STD_ZVAL(new_variable);

//
// set type and variable contents here
//

// introduce this variable by the name "new_variable_name" into the global symbol table
ZEND_SET_SYMBOL(&EG(symbol_table), "new_variable_name", new_variable);
```

The macro *ZEND_SET_SYMBOL* is now being called with a reference to the main, global symbol table by referring *EG(symbol_table)*.

Note: The *active_symbol_table* variable is a pointer, but *symbol_table* is not. This is why you have to use *EG(active_symbol_table)* and *&EG(symbol_table)* as parameters to *ZEND_SET_SYMBOL* - it requires a pointer.

Similarly, to get a more efficient version, you can hardcode the symbol table update:

```
zval *new_variable;

// allocate and initialize new container
MAKE_STD_ZVAL(new_variable);

//
// set type and variable contents here
//

// introduce this variable by the name "new_variable_name" into the global symbol table
zend_hash_update(
    &EG(symbol_table),
    "new_variable_name",
    strlen("new_variable_name") + 1,
    &new_variable,
    sizeof(zval *),
    NULL
);
```

Ejemplo 53-1 shows a sample source that creates two variables - *local_variable* with a local scope and *global_variable* with a global scope (see Figure 9.7). The full example can be found on the CD-ROM.

Note: You can see that the global variable is actually not accessible from within the function. This is

because it's not imported into the local scope using *global $global_variable;* in the PHP source.

Ejemplo 53-1. Creating variables with different scopes.

```
ZEND_FUNCTION(variable_creation)
{
    zval *new_var1, *new_var2;

    MAKE_STD_ZVAL(new_var1);
    MAKE_STD_ZVAL(new_var2);

    ZVAL_LONG(new_var1, 10);
    ZVAL_LONG(new_var2, 5);

    ZEND_SET_SYMBOL(EG(active_symbol_table), "local_variable", new_var1);
    ZEND_SET_SYMBOL(&EG(symbol_table), "global_variable", new_var2);

    RETURN_NULL();

}
```

Longs (Integers)

Now let's get to the assignment of data to variables, starting with longs. Longs are PHP's integers and are very simple to store. Looking at the *zval.value* container structure discussed earlier in this chapter, you can see that the long data type is directly contained in the union, namely in the *lval* field. The corresponding *type* value for longs is *IS_LONG* (see Ejemplo 53-2).

Ejemplo 53-2. Creation of a long.

```
zval *new_long;

MAKE_STD_ZVAL(new_long);

new_long->type = IS_LONG;
new_long->value.lval = 10;
```

Alternatively, you can use the macro *ZVAL_LONG*:

```
zval *new_long;

MAKE_STD_ZVAL(new_long);
ZVAL_LONG(new_long, 10);
```

Doubles (Floats)

Doubles are PHP's floats and are as easy to assign as longs, because their value is also contained directly in the union. The member in the *zval.value* container is *dval*; the corresponding type is *IS_DOUBLE*.

```
zval *new_double;

MAKE_STD_ZVAL(new_double);

new_double->type = IS_DOUBLE;
new_double->value.dval = 3.45;
```

Alternatively, you can use the macro *ZVAL_DOUBLE*:

```
zval *new_double;

MAKE_STD_ZVAL(new_double);
ZVAL_DOUBLE(new_double, 3.45);
```

Strings

Strings need slightly more effort. As mentioned earlier, all strings that will be associated with Zend's internal data structures need to be allocated using Zend's own memory-management functions. Referencing of static strings or strings allocated with standard routines is not allowed. To assign strings, you have to access the structure *str* in the *zval.value* container. The corresponding type is *IS_STRING*:

```
zval *new_string;
char *string_contents = "This is a new string variable";

MAKE_STD_ZVAL(new_string);

new_string->type = IS_STRING;
new_string->value.str.len = strlen(string_contents);
new_string->value.str.val = estrdup(string_contents);
```

Note the usage of Zend's **estrdup()** here. Of course, you can also use the predefined macro *ZVAL_STRING*:

```
zval *new_string;
char *string_contents = "This is a new string variable";

MAKE_STD_ZVAL(new_string);
ZVAL_STRING(new_string, string_contents, 1);
```

ZVAL_STRING accepts a third parameter that indicates whether the supplied string contents should be duplicated (using **estrdup()**). Setting this parameter to *1* causes the string to be duplicated; *0* simply uses the supplied pointer for the variable contents. This is most useful if you want to create a new variable referring to a string that's already allocated in Zend internal memory.

If you want to truncate the string at a certain position or you already know its length, you can use *ZVAL_STRINGL(zval, string, length, duplicate)*, which accepts an explicit string length to be set for the new string. This macro is faster than *ZVAL_STRING* and also binary-safe.

To create empty strings, set the string length to *0* and use *empty_string* as contents:

```
new_string->type = IS_STRING;
new_string->value.str.len = 0;
new_string->value.str.val = empty_string;
```

Of course, there's a macro for this as well (*ZVAL_EMPTY_STRING*):

```
MAKE_STD_ZVAL(new_string);
ZVAL_EMPTY_STRING(new_string);
```

Booleans

Booleans are created just like longs, but have the type *IS_BOOL*. Allowed values in *lval* are *0* and *1*:

```
zval *new_bool;

MAKE_STD_ZVAL(new_bool);

new_bool->type = IS_BOOL;
new_bool->value.lval = 1;
```

The corresponding macros for this type are *ZVAL_BOOL* (allowing specification of the value) as well as *ZVAL_TRUE* and *ZVAL_FALSE* (which explicitly set the value to *TRUE* and *FALSE*, respectively).

Arrays

Arrays are stored using Zend's internal hash tables, which can be accessed using the **zend_hash_*()** API. For every array that you want to create, you need a new hash table handle, which will be stored in the *ht* member of the *zval.value* container.

There's a whole API solely for the creation of arrays, which is extremely handy. To start a new array, you call **array_init()**.

```
zval *new_array;

MAKE_STD_ZVAL(new_array);

array_init(new_array);
```

array_init() always returns *SUCCESS*.

To add new elements to the array, you can use numerous functions, depending on what you want to do. Tabla 53-1, Tabla 53-2 and Tabla 53-3 describe these functions. All functions return *FAILURE* on failure and *SUCCESS* on success.

Tabla 53-1. Zend's API for Associative Arrays

Function	Description
add_assoc_long(zval *array, char *key, long n);()	Adds an element of type *long*.
add_assoc_unset(zval *array, char *key);()	Adds an unset element.
add_assoc_bool(zval *array, char *key, int b);()	Adds a Boolean element.
add_assoc_resource(zval *array, char *key, int r);()	Adds a resource to the array.

add_assoc_double (zval *array, char *key, double d);()	Adds a floating-point value.
add_assoc_string (zval *array, char *key, char *str, int duplicate);()	Adds a string to the array. The flag *duplicate* specifies whether the string contents have to be copied to Zend internal memory.
add_assoc_stringl (zval *array, char *key, char *str, uint length, int duplicate); ()	Adds a string with the desired length *length* to the array. Otherwise, behaves like **add_assoc_string()**.
add_assoc_zval (zval *array, char *key, zval *value);()	Adds a zval to the array. Useful for adding other arrays, objects, streams, etc...

Tabla 53-2. Zend's API for Indexed Arrays, Part 1

Function	Descriptio n

add_index_long (zval *array, uint idx, long n);()	Adds an element of type *long*.
add_index_unset (zval *array, uint idx); ()	Adds an unset element.
add_index_bool (zval *array, uint idx, int b);()	Adds a Boolean element.
add_index_resource(zval *array, uint idx, int r);()	Adds a resource to the array.
add_index_double (zval *array, uint idx, double d);()	Adds a floating-point value.
add_index_string (zval *array, uint idx, char *str, int duplicate);()	Adds a string to the array. The flag *duplicate* specifies whether the string contents have to be copied to Zend internal memory.

add_index_stringl(zval *array, uint idx, char *str, uint length, int duplicate);()	Adds a string with the desired length *length* to the array. This function is faster and binary-safe. Otherwise, behaves like **add_index_string()()**.
add_index_zval(zval *array, uint idx, zval *value);()	Adds a zval to the array. Useful for adding other arrays, objects, streams, etc...

Tabla 53-3. Zend's API for Indexed Arrays, Part 2

Function	Description
add_next_index_long(zval *array, long n);()	Adds an element of type *long*.
add_next_index_unset(zval *array);()	Adds an unset element.
add_next_index_bool(zval *array, int b);()	Adds a Boolean element.
add_next_index_resource(zval *array, int r);()	Adds a resource to the array.

add_next_index_double(zval *array, double d);()	Adds a floating-point value.
add_next_index_string(zval *array, char *str, int duplicate);()	Adds a string to the array. The flag *duplicate* specifies whether the string contents have to be copied to Zend internal memory.
add_next_index_stringl(zval *array, char *str, uint length, int duplicate);()	Adds a string with the desired length *length* to the array. This function is faster and binary-safe. Otherwise, behaves like **add_index_string()()**.
add_next_index_zval(zval *array, zval *value);()	Adds a zval to the array. Useful for adding other arrays, objects, streams, etc...

All these functions provide a handy abstraction to Zend's internal hash API. Of course, you can also use the hash functions directly - for example, if you already have a *zval* container allocated that you want to insert into an array. This is done using **zend_hash_update()()** for associative arrays (see Ejemplo 53-3) and **zend_hash_index_update()** for indexed arrays (see Ejemplo 53-4):

Ejemplo 53-3. Adding an element to an associative array.

```
zval *new_array, *new_element;
char *key = "element_key";

MAKE_STD_ZVAL(new_array);
MAKE_STD_ZVAL(new_element);

array_init(new_array);

ZVAL_LONG(new_element, 10);

if(zend_hash_update(new_array->value.ht, key, strlen(key) + 1, (void *)&new_element, si
{
    // do error handling here
}
```

Ejemplo 53-4. Adding an element to an indexed array.

```
zval *new_array, *new_element;
int key = 2;

MAKE_STD_ZVAL(new_array);
MAKE_STD_ZVAL(new_element);

array_init(new_array);

ZVAL_LONG(new_element, 10);

if(zend_hash_index_update(new_array->value.ht, key, (void *)&new_element, sizeof(zval *
{
    // do error handling here
}
```

To emulate the functionality of **add_next_index_*()**, you can use this:

```
zend_hash_next_index_insert(ht, zval **new_element, sizeof(zval *), NULL)
```

Note: To return arrays from a function, use **array_init()** and all following actions on the predefined variable *return_value* (given as argument to your exported function; see the earlier discussion of the call interface). You do not have to use *MAKE_STD_ZVAL* on this.

Tip: To avoid having to write *new_array->value.ht* every time, you can use *HASH_OF(new_array)*, which is also recommended for compatibility and style reasons.

Objects

Since objects can be converted to arrays (and vice versa), you might have already guessed that they have a lot of similarities to arrays in PHP. Objects are maintained with the same hash functions, but there's a different API for creating them.

To initialize an object, you use the function **object_init()**:

```
zval *new_object;

MAKE_STD_ZVAL(new_object);

if(object_init(new_object) != SUCCESS)
{
    // do error handling here
}
```

You can use the functions described in Tabla 53-4 to add members to your object.

Tabla 53-4. Zend's API for Object Creation

Function	Description
add_property_long(zval *object, char *key, long l);()	Adds a long to the object.
add_property_unset(zval *object, char *key);()	Adds an unset property to the object.
add_property_bool(zval *object, char *key, int b);()	Adds a Boolean to the object.
add_property_resource(zval *object, char *key, long r);()	Adds a resource to the object.
add_property_double(zval *object, char *key, double d);()	Adds a double to the object.
add_property_string(zval *object, char *key, char *str, int duplicate);()	Adds a string to the object.

add_property_stringl(zval *object, char *key, char *str, uint length, int duplicate);()	Adds a string of the specified length to the object. This function is faster than **add_property_string()** and also binary-safe.
add_property_zval(zval *obect, char *key, zval *container):()	Adds a *zval* container to the object. This is useful if you have to add properties which aren't simple types like integers or strings but arrays or other objects.

Resources

Resources are a special kind of data type in PHP. The term *resources* doesn't really refer to any special kind of data, but to an abstraction method for maintaining any kind of information. Resources are kept in a special resource list within Zend. Each entry in the list has a correspondending type definition that denotes the kind of resource to which it refers. Zend then internally manages all references to this resource. Access to a resource is never possible directly - only via a provided API. As soon as all references to a specific resource are lost, a corresponding shutdown function is called.

For example, resources are used to store database links and file descriptors. The *de facto* standard implementation can be found in the MySQL module, but other modules such as the Oracle module also make use of resources.

> **Nota:** In fact, a resource can be a pointer to anything you need to handle in your functions (e.g. pointer to a structure) and the user only has to pass a single resource variable to your function.

To create a new resource you need to register a resource destruction handler for it. Since you can store any kind of data as a resource, Zend needs to know how to free this resource if its not longer needed. This works by registering your own resource destruction handler to Zend which in turn gets called by Zend whenever your resource can be freed (whether manually or automatically). Registering your resource handler within Zend returns you the **resource type handle** for that resource. This handle is needed whenever you want to access a resource of this type later and is most of time stored in a global static variable within your extension. There is no need to worry about thread safety here because you only register your resource handler once during module initialization.

The Zend function to register your resource handler is defined as:

```
ZEND_API int zend_register_list_destructors_ex(rsrc_dtor_func_t ld, rsrc_dtor_func_t pl
```

There are two different kinds of resource destruction handlers you can pass to this function: a handler for normal resources and a handler for persistent resources. Persistent resources are for example used for database connection. When registering a resource, either of these handlers must be given. For the other handler just pass *NULL*.

zend_register_list_destructors_ex() accepts the following parameters:

ld	Normal resource destruction handler callback
pld	Pesistent resource destruction handler callback
type_name	A string specifying the name of your resource. It's always a good thing to specify an unique name within PHP for the resource type so when the user for example calls *var_dump ($resource);* he also gets the name of the resource.

module_number	The *module_number* is automatically available in your *PHP_MINIT_FUNCTION* function and therefore you just pass it over.

The return value is an unique integer ID for your **resource type**.

The resource destruction handler (either normal or persistent resources) has the following prototype:

```
void resource_destruction_handler(zend_rsrc_list_entry *rsrc TSRMLS_DC);
```

The passed *rsrc* is a pointer to the following structure:

```
typedef struct _zend_rsrc_list_entry {

    void *ptr;
    int type;
    int refcount;

} zend_rsrc_list_entry;
```

The member *void *ptr* is the actual pointer to your resource.

Now we know how to start things, we define our own resource we want register within Zend. It is only a simple structure with two integer members:

```
typedef struct {

    int resource_link;
    int resource_type;

} my_resource;
```

Our resource destruction handler is probably going to look something like this:

```
void my_destruction_handler(zend_rsrc_list_entry *rsrc TSRMLS_DC) {

    // You most likely cast the void pointer to your structure type

    my_resource *my_rsrc = (my_resource *) rsrc->ptr;

    // Now do whatever needs to be done with you resource. Closing
    // Files, Sockets, freeing additional memory, etc.
    // Also, don't forget to actually free the memory for your resource too!

    do_whatever_needs_to_be_done_with_the_resource(my_rsrc);
}
```

> **Nota:** One important thing to mention: If your resource is a rather complex structure which also contains pointers to memory you allocated during runtime you have to free them **before** freeing the resource itself!

Now that we have defined

1. what our resource is and

2. our resource destruction handler

we can go on and do the rest of the steps:

1. create a global variable within the extension holding the resource ID so it can be accessed from every function which needs it

2. define the resource name

3. write the resource destruction handler

4. and finally register the handler

```
// Somewhere in your extension, define the variable for your registered resources.
    // If you wondered what 'le' stands for: it simply means 'list entry'.
    static int le_myresource;

    // It's nice to define your resource name somewhere
    #define le_myresource_name  "My type of resource"

    [...]

    // Now actually define our resource destruction handler
    void my_destruction_handler(zend_rsrc_list_entry *rsrc TSRMLS_DC) {

        my_resource *my_rsrc = (my_resource *) rsrc->ptr;
        do_whatever_needs_to_be_done_with_the_resource(my_rsrc);
    }

    [...]

    PHP_MINIT_FUNCTION(my_extension) {

        // Note that 'module_number' is already provided through the
        // PHP_MINIT_FUNCTION() function definition.

        le_myresource = zend_register_resource_destructors_ex(my_destruction_handler, N

        // You can register additional resources, initialize
        // your global vars, constants, whatever.
    }
```

To actually register a new resource you use can either use the **zend_register_resource()** function or the **ZEND_REGISTER_RESOURE()** macro, both defined in zend_list.h . Although the arguments for both map 1:1 it's a good idea to always use macros to be upwards compatible:

```
int ZEND_REGISTER_RESOURCE(zval *rsrc_result, void *rsrc_pointer, int rsrc_type);
```

rsrc_result	This is an already initialized *zval** container.
rsrc_pointer	Your resource pointer you want to store.

rsrc_type	The type which you received when you registered the resource destruction handler. If you followed the naming scheme this would be *le_myresource*.

The return value is an unique integer identifier for that resource.

What is really going on when you register a new resource is it gets inserted in an internal list in Zend and the result is just stored in the given *zval* * container:

```
rsrc_id = zend_list_insert(rsrc_pointer, rsrc_type);

    if (rsrc_result) {
        rsrc_result->value.lval = rsrc_id;
        rsrc_result->type = IS_RESOURCE;
    }

    return rsrc_id;
```

The returned *rsrc_id* uniquly identifies the newly registered resource. You can use the macro *RETURN_RESOURE* to return it to the user:

```
RETURN_RESOURCE(rsrc_id)
```

> **Nota:** It is common practice that if you want to return the resource immidiately to the user you specify the *return_value* as the *zval* * container.

Zend now keeps track of all references to this resource. As soon as all references to the resource are lost, the destructor that you previously registered for this resource is called. The nice thing about this setup is that you don't have to worry about memory leakages introduced by allocations in your module - just register all memory allocations that your calling script will refer to as resources. As soon as the script decides it doesn't need them anymore, Zend will find out and tell you.

Now that the user got his resource, at some point he is passing it back to one of your functions. The *value.lval* inside the *zval* * container contains the key to your resource and thus can be used to fetch the resource with the following macro: *ZEND_FETCH_RESOURCE*:

```
ZEND_FETCH_RESOURCE(rsrc, rsrc_type, rsrc_id, default_rsrc_id, resource_type_name, resc
```

rsrc	This is your pointer which will point to your previously registered resource.

rsrc_type	This is the typecast argument for your pointer, e.g. *myresource **.
rsrc_id	This is the address of the *zval* *container the user passed to your function, e.g. *&z_resource* if *zval **z_resource* is given.
default_rsrc_id	This integer specifies the default resource *ID* if no resource could be fetched or -1.
resource_type_name	This is the name of the requested resource. It's a string and is used when the resource can't be found or is invalid to form a meaningful error message.
resource_type	The *resource_type* you got back when registering the resource destruction handler. In our example this was *le_myresource*.

This macro has no return value. It is for the developers convenience and takes care of TSRMLS arguments passing and also does check if the resource could be fetched. It throws a warning message and returns the current PHP function with *NULL* if there was a problem retrieving the resource.

To force removal of a resource from the list, use the function **zend_list_delete()**. You can also force the reference count to increase if you know that you're creating another reference for a previously allocated value (for example, if you're automatically reusing a default database link). For this case, use the function **zend_list_addref()**. To search for previously allocated resource entries, use **zend_list_find()**. The complete API can be found in `zend_list.h`.

Macros for Automatic Global Variable Creation

In addition to the macros discussed earlier, a few macros allow easy creation of simple global variables. These are nice to know in case you want to introduce global flags, for example. This is somewhat bad practice, but Table Tabla 53-5 describes macros that do exactly this task. They don't need any *zval* allocation; you simply have to supply a variable name and value.

Tabla 53-5. Macros for Global Variable Creation

Macro	Description
SET_VAR_STRING(name, value)	Creates a new string.
SET_VAR_STRINGL(name, value, length)	Creates a new string of the specified length. This macro is faster than *SET_VAR_STRING* and also binary-safe.
SET_VAR_LONG(name, value)	Creates a new long.
SET_VAR_DOUBLE(name, value)	Creates a new double.

Creating Constants

Zend supports the creation of true constants (as opposed to regular variables). Constants are accessed without the typical dollar sign (*$*) prefix and are available in all scopes. Examples include *TRUE* and *FALSE*, to name just two.

To create your own constants, you can use the macros in Tabla 53-6. All the macros create a constant with the specified name and value.

You can also specify flags for each constant:

- *CONST_CS* - This constant's name is to be treated as case sensitive.
- *CONST_PERSISTENT* - This constant is persistent and won't be "forgotten" when the current process carrying this constant shuts down.

To use the flags, combine them using a inary OR:

```
// register a new constant of type "long"
     REGISTER_LONG_CONSTANT("NEW_MEANINGFUL_CONSTANT", 324, CONST_CS |
     CONST_PERSISTENT);
```

There are two types of macros - *REGISTER_*_CONSTANT* and*REGISTER_MAIN_*_CONSTANT*. The first type creates constants bound to the current module. These constants are dumped from the symbol table as soon as the module that registered the constant is unloaded from memory. The second type creates constants that remain in the symbol table independently of the module.

Tabla 53-6. Macros for Creating Constants

Macro	Descriptio n
REGISTER_ LONG_CON STANT (name, value, flags) REGISTER_ MAIN_LON G_CONSTA NT(name, value, flags)	Registers a new constant of type long.
REGISTER_ DOUBLE_C ONSTANT (name, value, flags) REGISTER_ MAIN_DOU BLE_CONST ANT(name, value, flags)	Registers a new constant of type double.

REGISTER_STRING_CONSTANT (name, value, flags) *REGISTER_MAIN_STRING_CONSTANT(name, value, flags)*	Registers a new constant of type string. The specified string must reside in Zend's internal memory.
REGISTER_STRINGL_CONSTANT (name, value, length, flags) *REGISTER_MAIN_STRINGL_CONSTANT(name, value, length, flags)*	Registers a new constant of type string. The string length is explicitly set to *length*. The specified string must reside in Zend's internal memory.

Capítulo 54. Duplicating Variable Contents: The Copy Constructor

Sooner or later, you may need to assign the contents of one *zval* container to another. This is easier said than done, since the *zval* container doesn't contain only type information, but also references to places in Zend's internal data. For example, depending on their size, arrays and objects may be nested with lots of hash table entries. By assigning one *zval* to another, you avoid duplicating the hash table entries, using only a reference to them (at most).

To copy this complex kind of data, use the *copy constructor*. Copy constructors are typically defined in languages that support operator overloading, with the express purpose of copying complex types. If you define an object in such a language, you have the possibility of overloading the "=" operator, which is usually responsible for assigning the contents of the lvalue (result of the evaluation of the left side of the operator) to the rvalue (same for the right side).

Overloading means assigning a different meaning to this operator, and is usually used to assign a function call to an operator. Whenever this operator would be used on such an object in a program, this function would be called with the lvalue and rvalue as parameters. Equipped with that information, it can perform the operation it intends the "=" operator to have (usually an extended form of copying).

This same form of "extended copying" is also necessary for PHP's *zval* containers. Again, in the case of an array, this extended copying would imply re-creation of all hash table entries relating to this array. For strings, proper memory allocation would have to be assured, and so on.

Zend ships with such a function, called **zend_copy_ctor()** (the previous PHP ñalent was **pval_copy_constructor()**).

A most useful demonstration is a function that accepts a complex type as argument, modifies it, and then returns the argument:

```
zval *parameter;

if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "z", &parameter) == FAILURE)
   return;
}

// do modifications to the parameter here

// now we want to return the modified container:
*return_value == *parameter;
zval_copy_ctor(return_value);
```

The first part of the function is plain-vanilla argument retrieval. After the (left out) modifications, however, it gets interesting: The container of *parameter* is assigned to the (predefined) *return_value* container. Now, in order to effectively duplicate its contents, the copy constructor is called. The copy constructor works directly with the supplied argument, and the standard return values are *FAILURE* on failure and *SUCCESS* on success.

If you omit the call to the copy constructor in this example, both *parameter* and *return_value* would point to the same internal data, meaning that *return_value* would be an illegal additional reference to the same data structures. Whenever changes occurred in the data that *parameter* points to, *return_value* might be affected. Thus, in order to create separate copies, the copy constructor must be used.

The copy constructor's counterpart in the Zend API, the destructor **zval_dtor()**, does the opposite of the constructor.

Capítulo 55. Returning Values

Returning values from your functions to PHP was described briefly in an earlier section; this section gives the details. Return values are passed via the *return_value* variable, which is passed to your functions as argument. The *return_value* argument consists of a *zval* container (see the earlier discussion of the call interface) that you can freely modify. The container itself is already allocated, so you don't have to run *MAKE_STD_ZVAL* on it. Instead, you can access its members directly.

To make returning values from functions easier and to prevent hassles with accessing the internal structures of the *zval* container, a set of predefined macros is available (as usual). These macros automatically set the correspondent type and value, as described in Tabla 55-1 and Tabla 55-2.

> **Nota:** The macros in Tabla 55-1 automatically *return* from your function, those in Tabla 55-2 only *set* the return value; they don't return from your function.

Tabla 55-1. Predefined Macros for Returning Values from a Function

Macro	Descriptio n

RETURN_RESOURCE (resource)	Returns a resource.
RETURN_BOOL(bool)	Returns a Boolean.
RETURN_NULL()	Returns nothing (a NULL value).
RETURN_LONG(long)	Returns a long.
RETURN_DOUBLE (double)	Returns a double.
RETURN_STRING(string, duplicate)	Returns a string. The *duplicate* flag indicates whether the string should be duplicated using **estrdup()**.
RETURN_STRINGL (string, length, duplicate)	Returns a string of the specified length; otherwise, behaves like *RETURN_STRING*. This macro is faster and binary-safe, however.
RETURN_EMPTY_STRING()	Returns an empty string.
RETURN_FALSE	Returns Boolean false.
RETURN_TRUE	Returns Boolean true.

Tabla 55-2. Predefined Macros for Setting the Return Value of a Function

Macro	Description
RETVAL_RESOURCE(resource)	Sets the return value to the specified resource.
RETVAL_BOOL(bool)	Sets the return value to the specified Boolean value.
RETVAL_NULL	Sets the return value to NULL.
RETVAL_LONG(long)	Sets the return value to the specified long.
RETVAL_DOUBLE(double)	Sets the return value to the specified double.
RETVAL_STRING(string, duplicate)	Sets the return value to the specified string and duplicates it to Zend internal memory if desired (see also *RETURN_STRING*).

RETVAL_STRINGL (string, length, duplicate)	Sets the return value to the specified string and forces the length to become *length* (see also *RETVAL_STRING*). This macro is faster and binary-safe, and should be used whenever the string length is known.
RETVAL_EMPTY_STRING	Sets the return value to an empty string.
RETVAL_FALSE	Sets the return value to Boolean false.
RETVAL_TRUE	Sets the return value to Boolean true.

Complex types such as arrays and objects can be returned by using **array_init()** and **object_init()**, as well as the corresponding hash functions on *return_value*. Since these types cannot be constructed of trivial information, there are no predefined macros for them.

Capítulo 56. Printing Information

Often it's necessary to print messages to the output stream from your module, just as **print()** would be used within a script. PHP offers functions for most generic tasks, such as printing warning messages, generating output for **phpinfo()**, and so on. The following sections provide more details. Examples of these functions can be found on the CD-ROM.

---

zend_printf()

zend_printf() works like the standard **printf()**, except that it prints to Zend's output stream.

---

zend_error()

zend_error() can be used to generate error messages. This function accepts two arguments; the first is the error type (see `zend_errors.h`), and the second is the error message.

```
zend_error(E_WARNING, "This function has been called with empty arguments");
```

Tabla 56-1 shows a list of possible values (see Figura 56-1). These values are also referred to in `php.ini`. Depending on which error type you choose, your messages will be logged.

Tabla 56-1. Zend's Predefined Error Messages.

Error	Description
E_ERROR	Signals an error and terminates execution of the script immediately .
E_WARNING	Signals a generic warning. Execution continues.
E_PARSE	Signals a parser error. Execution continues.
E_NOTICE	Signals a notice. Execution continues. Note that by default the display of this type of error messages is turned off in `php.ini`.

E_CORE_ERROR	Internal error by the core; shouldn't be used by user-written modules.
E_COMPILE_ERROR	Internal error by the compiler; shouldn't be used by user-written modules.
E_COMPILE_WARNING	Internal warning by the compiler; shouldn't be used by user-written modules.

Figura 56-1. Display of warning messages in the browser.

Including Output in phpinfo()

After creating a real module, you'll want to show information about the module in **phpinfo()** (in addition to the module name, which appears in the module list by default). PHP allows you to create your own section in the **phpinfo()** output with the *ZEND_MINFO()* function. This function should be placed in the module descriptor block (discussed earlier) and is always called whenever a script calls **phpinfo()**.

PHP automatically prints a section in **phpinfo()** for you if you specify the *ZEND_MINFO* function, including the module name in the heading. Everything else must be formatted and printed by you.

Typically, you can print an HTML table header using **php_info_print_table_start()** and then use the standard functions **php_info_print_table_header()** and **php_info_print_table_row()**. As arguments, both take the number of columns (as integers) and the column contents (as strings). Ejemplo 56-1 shows a source example and its output. To print the table footer, use **php_info_print_table_end()**.

Ejemplo 56-1. Source code and screenshot for output in phpinfo().

```
php_info_print_table_start();
php_info_print_table_header(2, "First column", "Second column");
php_info_print_table_row(2, "Entry in first row", "Another entry");
php_info_print_table_row(2, "Just to fill", "another row here");
php_info_print_table_end();
```

Execution Information

You can also print execution information, such as the current file being executed. The name of the function currently being executed can be retrieved using the function **get_active_function_name()**. This function returns a pointer to the function name and doesn't accept any arguments. To retrieve the name of the file currently being executed, use **zend_get_executed_filename()**. This function accesses the executor globals, which are passed to it using the *TSRMLS_C* macro. The executor globals are automatically available to every function that's called directly by Zend (they're part of the *INTERNAL_FUNCTION_PARAMETERS* described earlier in this chapter). If you want to access the executor globals in another function that doesn't have them available automatically, call the macro *TSRMLS_FETCH()* once in that function; this will introduce them to your local scope.

Finally, the line number currently being executed can be retrieved using the function **zend_get_executed_lineno()**. This function also requires the executor globals as arguments. For examples of these functions, see Ejemplo 56-2.

Ejemplo 56-2. Printing execution information.

```
zend_printf("The name of the current function is %s<br>", get_active_function_name(TSRM
zend_printf("The file currently executed is %s<br>", zend_get_executed_filename(TSRMLS_
zend_printf("The current line being executed is %i<br>", zend_get_executed_lineno(TSRMl
```

Capítulo 57. Startup and Shutdown Functions

Startup and shutdown functions can be used for one-time initialization and deinitialization of your modules. As discussed earlier in this chapter (see the description of the Zend module descriptor block), there are module, and request startup and shutdown events.

The module startup and shutdown functions are called whenever a module is loaded and needs initialization; the request startup and shutdown functions are called every time a request is processed (meaning that a file is being executed).

For dynamic extensions, module and request startup/shutdown events happen at the same time.

Declaration and implementation of these functions can be done with macros; see the earlier section "Declaration of the Zend Module Block" for details.

Capítulo 58. Calling User Functions

You can call user functions from your own modules, which is very handy when implementing callbacks; for example, for array walking, searching, or simply for event-based programs.

User functions can be called with the function **call_user_function_ex()**. It requires a hash value for the function table you want to access, a pointer to an object (if you want to call a method), the function name, return value, number of arguments, argument array, and a flag indicating whether you want to perform zval separation.

```
ZEND_API int call_user_function_ex(HashTable *function_table, zval *object,
zval *function_name, zval **retval_ptr_ptr,
int param_count, zval **params[],
int no_separation);
```

Note that you don't have to specify both *function_table* and *object*; either will do. If you want to call a method, you have to supply the object that contains this method, in which case **call_user_function()**automatically sets the function table to this object's function table. Otherwise, you only need to specify *function_table* and can set *object* to *NULL*.

Usually, the default function table is the "root" function table containing all function entries. This function table is part of the compiler globals and can be accessed using the macro *CG*. To introduce the compiler globals to your function, call the macro *TSRMLS_FETCH* once.

The function name is specified in a *zval* container. This might be a bit surprising at first, but is quite a logical step, since most of the time you'll accept function names as parameters from calling functions within your script, which in turn are contained in *zval* containers again. Thus, you only have to pass your arguments through to this function. This *zval* must be of type *IS_STRING*.

The next argument consists of a pointer to the return value. You don't have to allocate memory for this container; the function will do so by itself. However, you have to destroy this container (using **zval_dtor()**) afterward!

Next is the parameter count as integer and an array containing all necessary parameters. The last argument specifies whether the function should perform zval separation - this should always be set to *0*. If set to *1*, the function consumes less memory but fails if any of the parameters need separation.

Ejemplo 58-1 shows a small demonstration of calling a user function. The code calls a function that's supplied to it as argument and directly passes this function's return value through as its own return value. Note the use of the constructor and destructor calls at the end - it might not be necessary to do it this way here (since they should be separate values, the assignment might be safe), but this is bulletproof.

Ejemplo 58-1. Calling user functions.

```
zval **function_name;
zval *retval;

if((ZEND_NUM_ARGS() != 1) || (zend_get_parameters_ex(1, &function_name) != SUCCESS))
{
    WRONG_PARAM_COUNT;
}

if((*function_name)->type != IS_STRING)
{
    zend_error(E_ERROR, "Function requires string argument");
}

TSRMSLS_FETCH();

if(call_user_function_ex(CG(function_table), NULL, *function_name, &retval, 0, NULL, 0)
{
    zend_error(E_ERROR, "Function call failed");
}

zend_printf("We have %i as type<br>", retval->type);

*return_value = *retval;
zval_copy_ctor(return_value);
zval_ptr_dtor(&retval);
```

```
<?php

dl("call_userland.so");

function test_function()
{

    print("We are in the test function!<br>");

    return("hello");

}

$return_value = call_userland("test_function");

print("Return value: \"$return_value\"<br>");
?>
```

Capítulo 59. Initialization File Support

PHP 4 features a redesigned initialization file support. It's now possible to specify default initialization entries directly in your code, read and change these values at runtime, and create message handlers for change notifications.

To create an .ini section in your own module, use the macros *PHP_INI_BEGIN()* to mark the beginning of such a section and *PHP_INI_END()* to mark its end. In between you can use *PHP_INI_ENTRY()* to create entries.

```
PHP_INI_BEGIN()
PHP_INI_ENTRY("first_ini_entry",  "has_string_value", PHP_INI_ALL, NULL)
PHP_INI_ENTRY("second_ini_entry", "2",                PHP_INI_SYSTEM, OnChangeSecond)
PHP_INI_ENTRY("third_ini_entry",  "xyz",              PHP_INI_USER, NULL)
PHP_INI_END()
```

The *PHP_INI_ENTRY()* macro accepts four parameters: the entry name, the entry value, its change permissions, and a pointer to a change-notification handler. Both entry name and value must be

specified as strings, regardless of whether they really are strings or integers.

The permissions are grouped into three sections:*PHP_INI_SYSTEM* allows a change only directly in the `php.ini` file; *PHP_INI_USER* allows a change to be overridden by a user at runtime using additional configuration files, such as `.htaccess`; and *PHP_INI_ALL* allows changes to be made without restrictions. There's also a fourth level, *PHP_INI_PERDIR*, for which we couldn't verify its behavior yet.

The fourth parameter consists of a pointer to a change-notification handler. Whenever one of these initialization entries is changed, this handler is called. Such a handler can be declared using the *PHP_INI_MH* macro:

```
PHP_INI_MH(OnChangeSecond);             // handler for ini-entry "second_ini_entry"

// specify ini-entries here

PHP_INI_MH(OnChangeSecond)
{

    zend_printf("Message caught, our ini entry has been changed to %s<br>", new_value);

    return(SUCCESS);

}
```

The new value is given to the change handler as string in the variable *new_value*. When looking at the definition of *PHP_INI_MH*, you actually have a few parameters to use:

```
#define PHP_INI_MH(name) int name(php_ini_entry *entry, char *new_value,
                                  uint new_value_length, void *mh_arg1,
                                  void *mh_arg2, void *mh_arg3)
```

All these definitions can be found in `php_ini.h`. Your message handler will have access to a structure that contains the full entry, the new value, its length, and three optional arguments. These optional arguments can be specified with the additional macros *PHP_INI_ENTRY1* (allowing one additional argument), *PHP_INI_ENTRY2* (allowing two additional arguments), and *PHP_INI_ENTRY3* (allowing three additional arguments).

The change-notification handlers should be used to cache initialization entries locally for faster access or to perform certain tasks that are required if a value changes. For example, if a constant connection to a certain host is required by a module and someone changes the hostname, automatically terminate the old connection and attempt a new one.

Access to initialization entries can also be handled with the macros shown in Tabla 59-1.

Tabla 59-1. Macros to Access Initialization Entries in PHP

Macro	Description
INI_INT (name)	Returns the current value of entry *name* as integer (long).

INI_FLT (name)	Returns the current value of entry *name* as float (double).
INI_STR (name)	Returns the current value of entry *name* as string. *Note:* This string is not duplicated, but instead points to internal data. Further access requires duplication to local memory.
INI_BOOL (name)	Returns the current value of entry *name* as Boolean (defined as *zend_bool*, which currently means *unsigned char*).
INI_ORIG_INT (name)	Returns the original value of entry *name* as integer (long).
INI_ORIG_FLT (name)	Returns the original value of entry *name* as float (double).

INI_ORIG_STR (name)	Returns the original value of entry *name* as string. Note: This string is not duplicated, but instead points to internal data. Further access requires duplication to local memory.
INI_ORIG_BOOL (name)	Returns the original value of entry *name* as Boolean (defined as *zend_bool*, which currently means *unsigned char*).

Finally, you have to introduce your initialization entries to PHP. This can be done in the module startup and shutdown functions, using the macros *REGISTER_INI_ENTRIES()* and *UNREGISTER_INI_ENTRIES()*:

```
ZEND_MINIT_FUNCTION(mymodule)
{

    REGISTER_INI_ENTRIES();

}

ZEND_MSHUTDOWN_FUNCTION(mymodule)
{

    UNREGISTER_INI_ENTRIES();

}
```

Capítulo 60. Where to Go from Here

You've learned a lot about PHP. You now know how to create dynamic loadable modules and statically linked extensions. You've learned how PHP and Zend deal with internal storage of variables and how you can create and access these variables. You know quite a set of tool functions that do a lot of routine tasks such as printing informational texts, automatically introducing variables to the symbol table, and so on.

Even though this chapter often had a mostly "referential" character, we hope that it gave you insight on how to start writing your own extensions. For the sake of space, we had to leave out a lot; we suggest that you take the time to study the header files and some modules (especially the ones in the `ext/standard` directory and the MySQL module, as these implement commonly known functionality). This will give you an idea of how other people have used the API functions - particularly those that didn't make it into this chapter.

Capítulo 61. Reference: Some Configuration Macros

`config.m4`

The file `config.m4` is processed by `buildconf` and must contain all the instructions to be executed during configuration. For example, these can include tests for required external files, such as header files, libraries, and so on. PHP defines a set of macros that can be used in this process, the most useful of which are described in Tabla 61-1.

Tabla 61-1. M4 Macros for `config.m4`

Macro	Description
AC_MSG_CHECKING(message)	Prints a "checking <message>" text during `configure`.
AC_MSG_RESULT(value)	Gives the result to *AC_MSG_CHECKING*; should specify either *yes* or *no* as *value*.
AC_MSG_ERROR(message)	Prints *message* as error message during `configure` and aborts the script.
AC_DEFINE(name,value,description)	Adds *#define* to `php_config.h` with the value of *value* and a comment that says *description* (this is useful for conditional compilation of your module).

AC_ADD_INCLUDE(path)	Adds a compiler include path; for example, used if the module needs to add search paths for header files.
AC_ADD_LIBRARY_WITH_PATH(libraryname,librarypath)	Specifies an additional library to link.
AC_ARG_WITH(modulename,description,unconditionaltest,conditionaltest)	Quite a powerful macro, adding the module with *description* to the `configure --help` output. PHP checks whether the option *--with-<modulename>* is given to the `configure` script. If so, it runs the script *unconditionaltest* (for example, *--with-myext=yes*), in which case the value of the option is contained in the variable *$withval*. Otherwise, it executes *conditionaltest*.

PHP_EXTENSION(modulename, [shared])	This macro is a *must* to call for PHP to configure your extension. You can supply a second argument in addition to your module name, indicating whether you intend compilation as a shared module. This will result in a definition at compile time for your source as *COMPILE_DL_<modulename>*.

Capítulo 62. API Macros

A set of macros was introduced into Zend's API that simplify access to *zval* containers (see Tabla 62-1).

Tabla 62-1. API Macros for Accessing *zval* Containers

Macro	Refers to
Z_LVAL(zval)	*(zval).value.lval*
Z_DVAL(zval)	*(zval).value.dval*
Z_STRVAL(zval)	*(zval).value.str.val*
Z_STRLEN(zval)	*(zval).value.str.len*
Z_ARRVAL(zval)	*(zval).value.ht*
Z_LVAL_P(zval)	*(*zval).value.lval*
Z_DVAL_P(zval)	*(*zval).value.dval*
Z_STRVAL_P(zval_p)	*(*zval).value.str.val*
Z_STRLEN_P(zval_p)	*(*zval).value.str.len*

Z_ARRVAL_P (zval_p)	*(*zval).value.ht*
Z_LVAL_PP (zval_pp)	*(**zval).value.lval*
Z_DVAL_PP (zval_pp)	*(**zval).value.dval*
Z_STRVAL_PP (zval_pp)	*(**zval).value.str.val*
Z_STRLEN_PP (zval_pp)	*(**zval).value.str.len*
Z_ARRVAL_PP (zval_pp)	*(**zval).value.ht*

VIII. PHP API: Interfaces para autores de extensiones

Tabla de contenidos

Capítulo 63. API de Secuencia para Autores de Extensiones PHP

Generalidades

La API de Secuencias de PHP introduce un enfoque unificado a la gestión de archivos y sockets en extensiones PHP. Usando una API única con funciones estándar para operaciones comunes, la API de secuencias le permite a su extensión acceder a archivos, sockets, URLs, memoria y objetos definidos por-script. Esta es una API extensible de tiempo de ejecución que permite la carga dinámica de módulos (¡y scripts!) para registrar nuevas secuencias.

El propósito de la API de secuencias es facilitar la apertura de archivos, URLs y otras fuentes de datos secuenciables a los desarrolladores, mediante una API unificada que es fácil de entender. La API se encuentra más o menos basada en la familia de funciones ANSI C stdio (con una semántica idéntica para la mayoría de sus funciones principales), así que los progamadores de C tendrán una sensación de familiaridad con las secuencias.

La API de secuencias opera en un par de niveles diferentes: al nivel base, la API define objetos php_stream para representar fuentes de datos secuenciables. En un nivel ligeramente más alto, la API define objetos php_stream_wrapper que "envuelven" la API de nivel más bajo para proveer soporte para la recuperación de datos y meta-datos desde URLs. Un parámetro de *contexto*

adicional, aceptado por la mayoría de funciones de creación de secuencia, es pasado al método *stream_opener* de la envoltura para configurar en detalle el comportamiento de la misma.

Cualquier secuencia, una vez abierta, puede tener también cualquier número de *filtros* aplicados sobre ella, los cuales procesan datos en la medida en que éstos son leídos desde/escritos hacia la secuencia.

Las secuencias pueden ser moldeadas (convertidas) a otros tipos de gestores de archivo, de modo que puedan ser usadas con bibliotecas externas sin mucho problema. Esto permite que aquellas bibliotecas accedan a datos directamente desde fuentes tipo URL. Si su sistema tiene la función **fopencookie()** o **funopen()**, ¡puede pasar incluso cualquier secuencia PHP a cualquier biblioteca que use stdio ANSI!

> **Nota:** Las funciones en este capítulo son para su uso en el código fuente de PHP y no son funciones de PHP. Las funciones de secuencias para usuarios pueden encontrarse en la Referencia de Secuencias.

Conceptos Básicos de Secuencias

El uso de secuencias es bastante similar al uso de funciones stdio ANSI. La principal diferencia está en el modo en que obtiene en un principo el gestor de la secuencia. En la mayoría de casos, usted usará **php_stream_open_wrapper()** para obtener el gestor de secuencia. Esta función trabaja de forma muy similar a fopen, y puede apreciarse en el siguiente ejemplo:

Ejemplo 63-1. ejemplo simple de secuencia que despliega la página de inicio de PHP

```
php_stream * stream = php_stream_open_wrapper("http://www.php.net", "rb", REPORT_ERRORS
if (stream) {
    while(!php_stream_eof(stream)) {
        char buf[1024];

        if (php_stream_gets(stream, buf, sizeof(buf))) {
            printf(buf);
        } else {
            break;
        }
    }
    php_stream_close(stream);
}
```

La tabla a continuación muestra los ñalentes de Secuencia a las funciones más comunes de stdio ANSI. A menos que se note lo contrario, las semánticas de las funciones son idénticas.

Tabla 63-1. Funciones ñalentes a stdio ANSI en la API de Secuencias

Función Stdio ANSI	Función de Secuencias PHP	Notas
fopen	php_stream_open_wrapper	Las secuencias incluyen parámetros opcionales
fclose	php_stream_close	
fgets	php_stream_gets	
fread	php_stream_read	Se asume que el parámetro nmemb tiene un valor de 1, así que el prototipo luce más como read(2)

Función Stdio ANSI	Función de Secuencias PHP	Notas
fwrite	php_stream_write	Se asume que el parámetro nmemb tiene un valor de 1, así que el prototipo luce más como write(2)
fseek	php_stream_seek	
ftell	php_stream_tell	
rewind	php_stream_rewind	
feof	php_stream_eof	
fgetc	php_stream_getc	
fputc	php_stream_putc	
fflush	php_stream_flush	
puts	php_stream_puts	La misma semántica que puts, NO fputs
fstat	php_stream_stat	Las secuencias tienen una estructura stat más rica

Las Secuencias como Recursos

Todas las secuencias son registradas como recursos cuando son creadas. Esto asegura que sean limpiadas apropiadamente incluso si ocurre un error fatal. Todas las funciones del sistema de archivos en PHP operan sobre recursos de secuencia - lo que quiere decir que sus extensiones pueden aceptar apuntadores de archivo PHP normales como parámetros también, y devolver secuencias desde sus funciones. La API de secuencias hace este proceso tan simple como es posible:

Ejemplo 63-2. Cómo aceptar una secuencia como parámetro

```
PHP_FUNCTION(ejemplo_escribir_hola)
{
    zval *zstream;
    php_stream *stream;

    if (FAILURE == zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "r", &zstream))
        return;

    php_stream_from_zval(stream, &zstream);

    /* ahora puede usar la secuencia. Sin embargo, usted no es el
       "duenyo" de la secuencia, lo es el script. Eso quiere decir que
       usted NO DEBE cerrar la secuencia, ya que producira un fallo en
       PHP! */

    php_stream_write(stream, "hola\n");

    RETURN_TRUE();
}
```

Ejemplo 63-3. Cómo devolver una secuencia desde una función

```
PHP_FUNCTION(ejemplo_abrir_pagina_php)
{
    php_stream *stream;

    stream = php_stream_open_wrapper("http://www.php.net", "rb", REPORT_ERRORS, NULL);

    php_stream_to_zval(stream, return_value);

    /* despues de este punto, el duenyo de la secuencia es el
       script. Si la cierra ahora, hara que PHP falle! */

}
```

Ya que las secuencias son limpiadas automáticamente, es tentador pensar que tenemos la oportunidad de ser programadores mediocres y no preocuparnos por cerrar las secuencias cuando hayamos terminado con ellas. Aunque tal enfoque podría funcionar, no es una buena idea por varias razones: las secuencias mantienen bloqueos sobre recursos del sistema mientras están abiertas, así que dejar un archivo abierto después que ha terminado de usarlo puede prevenir que otros procesos tengan acceso a los recursos. Si un script maneja un gran número de archivos, el conjunto de los recursos usados, tanto en términos de memoria y el exagerado número de archivos abiertos, puede producir que las peticiones al servidor web fallen. Suena mal, ¿no es así? La API de secuencias incluye algo de magia que le ayuda a mantener su código limpio - si una secuencia no es cerrada por su código cuando debería hacerlo, encontrará un poco de información útil de depuración en su registro de errores del servidor web.

> **Nota:** Siempre use una versión de PHP compilada para depuración cuando desarrolle una extensión (*--enable-debug* cuando ejecute configure), ya que se ha hecho un gran esfuerzo para advertirle sobre fugas de memoria y secuencias.

En algunos casos, es útil mantener una secuencia abierta durante la duración de una petición, para actuar como un registro o rastrear un archivo, por ejemplo. Escribir el código necesario para limpiar de manera segura ese tipo de secuencia no es difícil, pero se trata de varias líneas de código que no es escrictamente necesario. Para evitarse los problemas de escribir aquél código, puede marcar una secuencia para que sea apta para una auto-limpieza. Lo que quiere decir esto es que la API de secuencias no emitirá una advertencia cuando sea momento de auto-limpiar una secuencia. Para hacer esto, puede usar **php_stream_auto_cleanup()**.

Referencia de la API Común de Secuencias

Tabla de contenidos

php_stream_open_wrapper_as_file -- Abre una secuencia sobre un archivo o URL, y lo convierte a FILE*
php_stream_filter_register_factory -- Registra una fábrica de filtros con la API de Secuencias
php_stream_filter_unregister_factory -- Retira el registro de una fábrica de filtros con la API de secuencias

php_stream_stat_path

(no version information, might be only in CVS)

php_stream_stat_path -- Obtiene el status de un archivo o URL

Descripción

int **php_stream_stat_path** (char * ruta, php_stream_statbuf * ssb)

php_stream_stat_path() examina el archivo o URL especificado por *ruta* y devuelve información tal como el tamaño del archivo, los tiempos de acceso y creación y demás. El valor de retorno es 0 de tener éxito, 1 en caso de error. Para más información sobre la información devuelta, vea php_stream_statbuf.

php_stream_stat

(no version information, might be only in CVS)

php_stream_stat -- Obtiene el status del almacenamiento interno asociado con una secuencia

Descripción

int **php_stream_stat** (php_stream * secuencia, php_stream_statbuf * ssb)

php_stream_stat() examina el almacenamiento al que está sujeto la *secuencia*, y devuelve información como el tamaño de archivo, las fechas de acceso y creación y demás. El valor de retorno es 0 en caso de éxito, -1 si ocurre un error. Para más información sobre la información devuelta, vea php_stream_statbuf.

php_stream_open_wrapper

(no version information, might be only in CVS)

php_stream_open_wrapper -- Abre una secuencia sobre un archivo o URL

Descripción

php_stream * **php_stream_open_wrapper** (char * ruta, char * modo, int opciones, char ** abierto)

php_stream_open_wrapper() abre una secuencia sobre el archivo, URL, u otro recurso envuelto indicado por *ruta*. Dependiendo del valor de *modo*, la secuencia puede ser abierta para lectura, escritura, adición o alguna combinación de éstos modos. Vea la tabla más adelante para conocer los

diferentes modos que pueden ser usados; adicionalmente a los caracteres listados más abajo, puede incluir el caracter 'b' ya sea como el segundo o último caracter en la cadena de modo. La presencia del caracter 'b' le informa a la implementación de secuencias en cuestión que abra la secuencia en modo seguro con material binario.

El caracter 'b' es ignorado en todos los sistemas compatibles con POSIX, que tratan los archivos binarios y de texto en la misma manera. Es una buena idea especificar el caracter 'b' cuando su secuencia trabaje con datos en donde todos los 8 bits son importantes, de modo que su código funcione cuando sea compilado en un sistema en donde la bandera 'b' es importante.

Cualquier archivo local creado por la API de secuencias tendrá sus permisos iniciales definidos de acuerdo a los valores predeterminados del sistema operativo - bajo sistemas basados en Unix esto quiere decir que se usará el valor umask del proceso. Bajo Windows, el dueño del archivo será el proceso que lo originó. Todo archivo remoto será creado de acuerdo a la envoltura de URL que fuera usada para abrir el archivo, y las credenciales entregadas al servidor remoto.

`r`

Abre un archivo de texto para lectura. La secuencia es ubicada al comienzo del archivo.

`r+`

Abre un archivo de texto para lectura y escritura. La secuencia es ubicada el comienzo del archivo.

`w`

Trunca el archivo a una longitud de cero, o crea el archivo de texto para escritura. La secuencia es ubicada al comienzo del archivo.

`w+`

Abre un archivo de texto para lectura y escritura. El archivo es creado si no existe, o de otra forma es truncado. La secuencia es ubicada al comienzo del archivo.

`a`

Abre para escritura. El archivo es creado si no existe. La secuencia es ubicada al final del archivo.

`a+`

Abre un archivo de texto para lectura y escritura. El archivo es creado si no existe. La secuencia es ubicada al final del archivo.

opciones afecta el modo en que se interpreta la ruta/URL de la secuencia, los chequeos de modo seguro y las acciones tomadas si hay un error durante la apertura de la secuencia. Vea Opciones de apertura de secuencia para más información sobre las opciones.

Si *abierto* es diferente a NULL, éste parámetro será definido como una cadena que contiene el nombre del archivo/recurso real que fue abierto. Esto es importante cuando las opciones incluyen **`USE_PATH`**, valor que causa que include_path sea usado en busca del archivo. Usted, el origen de la llamada, es responsable de llamar **efree()** sobre el nombre de archivo devuelto en este parámetro.

Nota: Si ha especificado **`STREAM_MUST_SEEK`** en *opciones*, la ruta devuelta en *abierto* puede no ser el nombre de la secuencia real que le fue devuelta. Sin embargo, será el nombre del recurso original desde el cual se manufacturó la secuencia sensible a búsquedas.

php_stream_read

(no version information, might be only in CVS)

php_stream_read -- Leer un número de bytes desde una secuencia a un búfer

Descripción

size_t **php_stream_read** (php_stream * secuencia, char * buf, size_t conteo)

php_stream_read() lee hasta *conteo* bytes de datos desde *secuencia* y los copia en el búfer *buf*.

php_stream_read() devuelve el número de bytes que fueron leídos satisfactoriamente. No hay distinción entre una lectura fallida o una condición de final-de-archivo - use **php_stream_eof()** para revisar si se encuentra en **`EOF`**.

La posición interna de la secuencia es desplazada hacia adelante por el número de bytes que fueron leídos, de modo que las lecturas subsiguientes continuarán desde ese punto.

Si hay menos de *conteo* bytes disponibles para su lectura, ésta llamada creará un bloqueo (esperará) hasta que el número requerido se encuentre disponible, dependiendo del status de bloqueo de la secuencia. Por defecto, una secuencia es abierta en modo de bloqueo. Cuando se lee desde archivos regulares, el modo de bloqueo usualmente no representará ninguna diferencia: cuando la secuencia alcance el **`EOF`**, **php_stream_read()** devolverá un valor menor que *conteo*, y 0 en cualquier lectura subsiguiente.

php_stream_write

(no version information, might be only in CVS)

php_stream_write -- Escribir un númedo de bytes desde un búfer a una secuencia

Descripción

size_t **php_stream_write** (php_stream * secuencia, const char * buf, size_t conteo)

php_stream_write() escribe *conteo* bytes de datos desde *buf* en *secuencia*.

php_stream_write() devuelve el número de bytes que fueron escritos satisfactoriamente. Si hubo un error, el número de bytes escritos será menor que *conteo*.

La posición interna de la secuencia es desplazada hacia adelante en el número de bytes que fueron escritos, de modo que cualquier escritura subsiguiente continuará desde ese punto.

php_stream_eof

(no version information, might be only in CVS)

php_stream_eof -- Chequear por una condición de fin-de-archivo en una secuencia

Descripción

int **php_stream_eof** (php_stream * secuencia)

php_stream_eof() chequea por una condición de fin-de-archivo en *secuencia*.

php_stream_eof() devuelve 1 para indicar **EOF**, 0 si no hay **EOF** y -1 para indicar un error.

php_stream_getc

(no version information, might be only in CVS)

php_stream_getc -- Leer un byte único desde una secuencia

Descripción

int **php_stream_getc** (php_stream * secuencia)

php_stream_getc() lee un caracter único desde *secuencia* y lo devuelve como un char sin signo, moldeado como un int, o **EOF** si se ha llegado al fin-de-archivo, u ocurrió un error.

php_stream_getc() puede bloquear en el mismo modo en que bloquea **php_stream_read()**.

La posición interna de la secuencia es desplazada hacia adelante en 1 si tiene éxito.

php_stream_gets

(no version information, might be only in CVS)

php_stream_gets -- Leer una línea de datos desde una secuencia a un búfer

Descripción

char * **php_stream_gets** (php_stream * secuencia, char * buf, size_t conteo)

php_stream_gets() lee hasta *conteo*-1 bytes de datos desde *secuencia* y los copia en el búfer *buf*. La lectura se detiene luego de un **EOF** o una nueva línea. Si una nueva línea es leída, ésta es almacenada en *buf* como parte de los datos devueltos. Un caracter de terminación NUL es almacenado como el último caracter en el búfer.

php_stream_read() devuelve *buf* de tener éxito, o NULL de lo contrario.

La posición interna de la secuencia es desplazada hacia adelante en el número de bytes que fueron

leídos, de modo que cualquier lectura subsiguiente continuará a partir de ese punto.

Esta función puede bloquear en el mismo modo en que lo hace **php_stream_read()**.

php_stream_close

(no version information, might be only in CVS)

php_stream_close -- Cerrar una secuencia

Descripción

int **php_stream_close** (php_stream * secuencia)

php_stream_close() cierra de forma segura la *secuencia* y libera los recursos asociados con ella. Luego de que la *secuencia* ha sido cerrada, su valor es indefinido y no debe ser usado.

php_stream_close() devuelve 0 si la secuencia fue cerrada, o **`EOF`** para indicar un error. Independientemente del éxito de la llamada, *secuencia* se hace indefinida y no debe ser usada después de una llamada a esta función.

php_stream_flush

(no version information, might be only in CVS)

php_stream_flush -- Volcar los búferes de secuencia a la unidad de almacenamiento

Descripción

int **php_stream_flush** (php_stream * secuencia)

php_stream_flush() produce que cualquier dato retenido en los búferes de escritura de la *secuencia* sean aplicados a la unidad de almacenamiento interna.

php_stream_flush() devuelve 0 si los búferes fueron volcados, o si los éstos no necesitaban ser volcados, y devuelve **`EOF`** para indicar un error.

php_stream_seek

(no version information, might be only in CVS)

php_stream_seek -- Reubicar una secuencia

Descripción

int **php_stream_seek** (php_stream * secuencia, off_t desplazamiento, int a_partir_de)

php_stream_seek() reubica la posición interna de *secuencia*. La nueva posición es determinada al sumar el *desplazamiento* a la posición indicada por *a_partir_de*. Si *a_partir_de* es definido como

SEEK_SET, **SEEK_CUR** o **SEEK_END**, el desplazamiento es relativo al comienzo de la secuencia, la posición actual o el final de la secuencia, respectivamente.

php_stream_seek() devuelve 0 de tener éxito, y -1 si ocurrió un error.

> **Nota:** No todas las secuencias soportan la reubicación, aunque la API de secuencias emulará la operación si *a_partir_de* es definido como **SEEK_CUR** y *desplazamiento* es un valor positivo, llamando **php_stream_read()** para leer (y descartar) tantos bytes como el valor de *desplazamiento*.
>
> La emulación es aplicada únicamente cuando la implementación interna de la secuencia no soporta la reubicación. Si la secuencia es (por ejemplo) una secuencia basada en archivos que se encuentra envolviendo un pipe no-reubicable, la api de secuencias no aplicará la emulación ya que la secuencia basada en archivos implementa la operación; la reubicación fallará y se devolverá un resultado de error a la función que hace la llamada.

php_stream_tell

(no version information, might be only in CVS)

php_stream_tell -- Determinar la posición de una secuencia

Descripción

off_t **php_stream_tell** (php_stream * secuencia)

php_stream_tell() devuelve la posición interna de *secuencia*, relativa al comienzo de la misma. Si ocurre un error, el valor -1 es devuelto.

php_stream_copy_to_stream

(no version information, might be only in CVS)

php_stream_copy_to_stream -- Copiar datos de una secuencia a otra

Descripción

size_t **php_stream_copy_to_stream** (php_stream * fuente, php_stream * destino, size_t long_max)

php_stream_copy_to_stream() intenta leer hasta *long_max* bytes de datos desde *fuente* y los escribe en *destino*, y devuelve el número de bytes que fueron copiados satisfactoriamente.

Si desea copiar todos los datos restantes de la secuencia *fuente*, pase la constante **PHP_STREAM_COPY_ALL** como el valor de *long_max*.

> **Nota:** Esta función intentará copiar los datos en la manera más eficiente posible, usando archivos referenciados en memoria cuando sea posible.

php_stream_copy_to_mem

(no version information, might be only in CVS)

php_stream_copy_to_mem -- Copiar datos de una secuencia en un búfer reservado

Descripción

size_t **php_stream_copy_to_mem** (php_stream * fuente, char ** buf, size_t long_max, int persistente)

php_stream_copy_to_mem() reserva un búfer de *long_max*+1 bytes de longitud usando **pemalloc ()** (pasando *persistente*). Luego lee *long_max* bytes desde *fuente* y los almacena en el búfer reservado.

El búfer reservado es devuelto en *buf*, y el número de bytes leídos satisfactoriamente. Usted, el origen de la llamada, es responsable de liberar el búfer, pasando éste y el parámetro *persistente* a **pefree()**.

Si desea copiar todos los datos restantes de la secuencia *fuente*, pase la constante **PHP_STREAM_COPY_ALL** como el valor de *long_max*.

> **Nota:** Esta función intentará copiar los datos en la manera más eficiente posible, usando archivos referenciados en memoria cuando sea posible.

php_stream_make_seekable

(no version information, might be only in CVS)

php_stream_make_seekable -- Convertir una secuencia a una secuencia reubicable

Descripción

int **php_stream_make_seekable** (php_stream * secuencia_origen, php_stream ** secuencia_nueva, int banderas)

php_stream_make_seekable() chequea si *secuencia_origen* es reubicable. Si no lo es, copiará los datos en una nueva secuencia temporal. Si tiene éxito, *secuencia_nueva* es siempre definida como la secuencia de uso válido, incluso cuando la secuencia original era reubicable.

banderas le permite especificar sus preferencias para la secuencia reubicable que es devuelta: use **PHP_STREAM_NO_PREFERENCE** para usar la secuencia reubicable predeterminada (que usa un búfer de memoria expansible dinámicamente, pero cambia a un tipo de almacenamiento respaldado en un archivo temporal cuando el tamaño de la secuencia se hace más largo), o use **PHP_STREAM_PREFER_STDIO** para indicar el tipo de almacenamiento "regular", respaldado en un archivo temporal.

Tabla 63-1. Valores de retorno de php_stream_make_seekable()

Valor	Significado
PHP_STREAM_UNCHANGED	La secuencia original era reubicable en cualquier caso. *secuencia_nueva* es definida al valor de *seuencia_original*.
PHP_STREAM_RELEASED	La secuencia original no era reubicable y ha sido liberada. *secuencia_nueva* es definida como la nueva secuencia reubicable. Usted no debe usar más valor de *secuencia_original*.
PHP_STREAM_FAILED	Un error ocurrió mientras se intentaba la conversión. *secuencia_nueva* se define como NULL; *secuencia_original* sigue siendo válida.
PHP_STREAM_CRITICAL	Un error ocurrión durante el intento de conversión que ha dejado a *secuencia_original* en un estado indeterminado. *secuencia_nueva* se define como NULL y es altamente recomendable que usted cierre *secuencia_original*.

Nota: Si necesita reubicar y escribir a la secuencia, no tiene sentido que use esta función, ya que no se garantiza que la secuencia que devuelve esté asociada con el mismo recurso que la secuencia original.

Nota: Si sólo necesita hacer búsquedas de posición hacia adelante, no hay necesidad de llamar esta función, ya que la API de secuencias emulará las reubicaciones hacia adelante cuando el parámetro a_partir_de es **`SEEK_CUR`**.

Nota: Si *secuencia_original* es basada en red, esta función creará un bloqueo hasta que los contenidos completos hayan sido descargados.

Nota: ¡NUNCA llame esta función con una *secuencia_original* que sea una referencia por un apuntador de archivo en un script PHP! ¡Esta función puede causar que la secuencia interna sea cerrada, cosa que causaría un fallo en el programa cuando el script acceda nuevamente al apuntador de archivo!

Nota: En muchos casos, esta función sólo puede tener éxito cuando *secuencia_original* es una secuencia abierta recientemente sin datos en búfer en la capa de la secuencia. Por esa razón, y dado que es difícial usar correctamente esta función, es recomendable que use **php_stream_open_wrapper()** y pase el valor **`PHP_STREAM_MUST_SEEK`** en sus opciones en lugar de llamar esta función directamente.

php_stream_cast

(no version information, might be only in CVS)

php_stream_cast -- Convertir una secuencia a otra forma, como un FILE* o un socket

Descripción

int **php_stream_cast** (php_stream * secuencia, int moldear_como, void ** ret, int banderas)

php_stream_cast() intenta convertir *secuencia* a un recurso indicado por *moldear_como*. Si *ret* es NULL, la secuencia es consultada para conocer si tal conversión es posible, sin efectuar realmente la conversión (sin embargo, algún estado interno de secuencias *puede* ser modificado en este caso). Si *banderas* es definido como **`REPORT_ERRORS`**, se mostrará un mensaje de error si hay un error durante la conversión.

Nota: Esta función devuelve **SUCCESS** en caso de tener éxito, o **FAILURE** si falla. Advierta que usted debe comparar explícitamente el valor de retorno con **SUCCESS** o **FAILURE** debido a los valores internos de éstas constantes. Una simple expresión booleana no será interpretada como usted puede esperar.

Tabla 63-1. Tipos de recurso para *moldear_como*

Valor	Significado
PHP_STREAM_AS_STDIO	Solicita un FILE* ANSI que represente la secuencia
PHP_STREAM_AS_FD	Solicita un descriptor de archivo POSIX que represente la secuencia
PHP_STREAM_AS_SOCKET D	Solicita un descriptor de socket de red que represente la secuencia

Adicionalmente a los tipos de recurso básicos mencionados anteriormente, el proceso de conversión puede ser alterado mediante el uso de las siguientes banderas, al usar el operador OR para combinar el tipo de recurso con uno o más de los siguientes valores:

Tabla 63-2. Tipos de recurso para *moldear_como*

Valor	Significado
PHP_STREAM_ CAST_TRY_H ARD	Intenta lo mejor posible, al precio de recursos adicionales, para asegurarse de que la conversión tenga éxito
PHP_STREAM_ CAST_RELEAS E	Le informa a la API de secuencias que otra fuente de código (posiblemente una biblioteca de algún tercero) será responsable de cerrar el recurso/gestor interno. Esto causa que la *secuencia* sea cerrada de forma tal que el gestor interno es conservado y devuelto en *ret*. Si esta función tiene éxito, la *secuencia* debe ser considerada cerrada y no debe ser usada más.

Nota: Si su sistema soporta el uso de **fopencookie()** (sistemas con glibc 2 o versiones posteriores), la API de secuencias será capaz siempre de sintetizar un apuntador FILE* ANSI sobre cualquier secuencia. Aunque esto es tremendamente útil para pasar cualquier secuencia PHP a cualquier biblioteca de terceros, tal comportamiento no es portable. Es necesario que considere las implicaciones de portabilidad antes de distribuir su extensión. Si la síntesis fopencookie no es deseable, debe consultar a la secuencia para ver si ésta soporta FILE* de forma natural mediante el uso de **php_stream_is()**

Nota: Si solicita una secuencia basada en sockets para un FILE*, la API de secuencias usará **fdopen()** para crearlo por usted. Advierta que hacer esto puede causar la pérdida de datos que fueran puestos en búfer en la capa de secuencias si usted entremezcla las llamadas de la API de secuencias con llamadas stdio ANSI.

Vea también **php_stream_is()** y **php_stream_can_cast()**.

php_stream_can_cast

(no version information, might be only in CVS)

php_stream_can_cast -- Determina si una secuencia puede ser convertida en otra forma, como un FILE* o un socket

Descripción

int **php_stream_can_cast** (php_stream * secuencia, int moldear_como)

Esta función es ñalente a llamar **php_stream_cast()** con *ret* definido como NULL y *banderas* definido como 0. Devuelve **SUCCESS** si la secuencia puede ser convertida a la forma solicitada, o **FAILURE** si la conversión no puede ser efectuada.

> **Nota:** Aunque esta función no realizará la conversión, algún estado interno de secuencias *puede* ser modificado por esta llamada.
>
> **Nota:** Usted debe comparar explícitamente el valor de retorno de ésta función con una de las constantes, tal y como se describe en **php_stream_cast()**.

Vea también **php_stream_cast()** y **php_stream_is()**.

php_stream_is_persistent

(no version information, might be only in CVS)

php_stream_is_persistent -- Determina si una secuencia es persistente

Descripción

int **php_stream_is_persistent** (php_stream * secuencia)

php_stream_is_persistent() devuelve 1 si la secuencia es persistente, o 0 de lo contrario.

php_stream_is

(no version information, might be only in CVS)

php_stream_is -- Determina si una secuencia es de un tipo particular

Descripción

int **php_stream_is** (php_stream * secuencia, int es_del_tipo)

php_stream_is() devuelve 1 si *secuencia* es del tipo especificado por *es_del_tipo*, o 0 de lo contrario.

Tabla 63-1. Valores para *es_del_tipo*

Valor	Significado
PHP_STREAM_IS_STDIO	La secuencia es implementada usando stdio
PHP_STREAM_IS_SOCKET	La secuencia es implementada usando un socket de red

Valor	Significado
PHP_STREAM_IS_USERSPACE	La secuencia es implementada usando un objeto de espacio de usuario
PHP_STREAM_IS_MEMORY	La secuencia es implementada usando una secuencia con memoria que crece a medida que es solicitada

Nota: Las "constantes" PHP_STREAM_IS_XXX son definidas en realidad como apuntadores a la estructura interna de operaciones de secuencia. Si su extensión (o alguna otra extensión) define secuencias adicionales, la extensión debe declarar también una constante PHP_STREAM_IS_XXX en su archivo de cabecera, que usted pueda usar como base para esta comparación.

Nota: Esta función es implementada como una sencilla (y rápida) comparación de apuntadores, y no modifica el estado de la secuencia en forma alguna.

Vea también **php_stream_cast()** y **php_stream_can_cast()**.

php_stream_passthru

(no version information, might be only in CVS)

php_stream_passthru -- Imprime todos los datos restantes de una secuencia

Descripción

size_t **php_stream_passthru** (php_stream * secuencia)

php_stream_passthru() imprime todos los datos restantes de *secuencia* en el búfer de salida activo y devuelve el número de bytes escritos. Si el uso de búferes está deshabilitado, los datos son escritos directamente a la salida, la cual se refiere al navegador que hace la petición en el caso de PHP sobre un servidor web, o stdout para código PHP basado en CLI. Esta función usará archivos referenciados en memoria si es posible para mejorar el rendimiento.

php_register_url_stream_wrapper

(no version information, might be only in CVS)

php_register_url_stream_wrapper -- Registra una envoltura con la API de Secuencias

Descripción

int **php_register_url_stream_wrapper** (char * protocolo, php_stream_wrapper * envoltura, TSRMLS_DC)

php_register_url_stream_wrapper() registra *envoltura* como el gestor para el protocolo especificado por *protocolo*.

Nota: Si llama esta función desde un módulo cargable, usted *DEBE* llamar **php_unregister_url_stream_wrapper()** en su función de cierre del módulo, o de otro modo PHP producirá un fallo.

php_unregister_url_stream_wrapper

(no version information, might be only in CVS)

php_unregister_url_stream_wrapper -- Retira el registro de una envoltura de la API de Secuencias

Descripción

int **php_unregister_url_stream_wrapper** (char * protocolo, TSRMLS_DC)

php_unregister_url_stream_wrapper() retira el registro de la envoltura asociada con *protocolo*.

php_stream_open_wrapper_ex

(no version information, might be only in CVS)

php_stream_open_wrapper_ex -- Abre una secuencia sobre un archivo o URL, especificando el contexto

Descripción

php_stream * **php_stream_open_wrapper_ex** (char * ruta, char * modo, int opciones, char ** abierto, php_stream_context * contexto)

php_stream_open_wrapper_ex() es exactamente como **php_stream_open_wrapper()**, pero le permite especificar un objeto php_stream_context usando *contexto*. Para conocer más sobre los contextos de secuencia, vea Contextos de Secuencia.

php_stream_open_wrapper_as_file

(no version information, might be only in CVS)

php_stream_open_wrapper_as_file -- Abre una secuencia sobre un archivo o URL, y lo convierte a FILE*

Descripción

FILE * **php_stream_open_wrapper_as_file** (char * ruta, char * modo, int opciones, char ** abierto)

php_stream_open_wrapper_as_file() es exactamente como **php_stream_open_wrapper()**, pero convierte la secuencia a un FILE* ANSI de stdio y devuelve éste apuntador en lugar de la secuencia. Este es un atajo conveniente para extensiones que pasan un valor FILE* a las bibliotecas de terceros.

php_stream_filter_register_factory

(no version information, might be only in CVS)

php_stream_filter_register_factory -- Registra una fábrica de filtros con la API de Secuencias

Descripción

int **php_stream_filter_register_factory** (const char * patron_filtros, php_stream_filter_factory * fabrica)

Use esta función para registrar una fábrica de filtros con el nombre dado por *patron_filtros*. *patron_filtros* puede ser un nombre de cadena normal (como *mi_filtro*) o un patrón global (es decir, *mi_clase_de_filtro.**) para permitir que un filtro único realice diferentes operaciones dependiendo del nombre exacto del filtro invocado (p.ej. *mi_clase_de_filtro.foo*, *mi_clase_de_filtro.bar*, etc...)

> **Nota:** Los filtros registrados por una extensión cargable deben asegurarse de llamar php_stream_filter_unregister_factory() durante MSHUTDOWN.

php_stream_filter_unregister_factory

(no version information, might be only in CVS)

php_stream_filter_unregister_factory -- Retira el registro de una fábrica de filtros con la API de secuencias

Descripción

int **php_stream_filter_unregister_factory** (const char * patron_filtros)

Retira el registro de la *fabrica_de_filtros* especificada por el *patron_filtros*, de modo que no sigue estando disponible para su uso.

> **Nota:** Los filtros registrados por una extensión cargable deben asegurarse de llamar php_stream_filter_unregister_factory() durante MSHUTDOWN.

Referencia de API de Directorios de Secuencias

Tabla de contenidos

Las funciones listadas en esta sección trabajan sobre archivos locales, asi como sobre archivos remotos (¡provisto que la envoltura soporta esta funcionalidad!).

php_stream_opendir

(no version information, might be only in CVS)

php_stream_opendir -- Abre un directorio para la enumeración de archivos

Descripción

php_stream * **php_stream_opendir** (char * ruta, php_stream_context * contexto)

php_stream_opendir() devuelve una secuencia que puede ser usada para listar los archivos contenidos en el directorio especificado por *ruta*. Esta función es ñalente en la práctica a la función POSIX **opendir()**. Aunque ésta función devuelve un objeto php_stream, no se recomienda usar las funciones de la API común sobre éstas secuencias.

php_stream_readdir

(no version information, might be only in CVS)

php_stream_readdir -- Recuperar la siguiente entrada de directorio desde un directorio abierto

Descripción

php_stream_dirent * **php_stream_readdir** (php_stream * secuencia_dir, php_stream_dirent * ent)

php_stream_readdir() lee la siguiente entrada de directorio desde *secuencia_dir* y la almacena en *ent*. Si la función tiene éxito, el valor de retorno es *ent*. Si la función falla, el valor de retorno es NULL. Vea php_stream_dirent para más detalles sobre la información devuelta por cada entrada de directorio.

php_stream_rewinddir

(no version information, might be only in CVS)

php_stream_rewinddir -- Restablecer una secuencia de directorio a la primera entrada

Descripción

int **php_stream_rewinddir** (php_stream * secuencia_dir)

php_stream_rewinddir() restablece la secuencia de directorio a la primera entrada. Devuelve 0 en caso de tener éxito, o -1 de encontrar un fallo.

php_stream_closedir

(no version information, might be only in CVS)

php_stream_closedir -- Cerrar una secuencia de directorio y liberar los recursos

Descripción

int **php_stream_closedir** (php_stream * secuencia_dir)

php_stream_closedir() cierra una secuencia de directorio y libera los recursos asociados con ella. Devuelve 0 de tener éxito, y -1 en caso de fallo.

Referencia de API de Archivos de Secuencias

Tabla de contenidos

php_stream_fopen_from_file

(no version information, might be only in CVS)

php_stream_fopen_from_file -- Convertir un FILE* ANSI a una secuencia

Descripción

php_stream * **php_stream_fopen_from_file** (FILE * archivo, char * modo)

php_stream_fopen_from_file() devuelve una secuencia basada en el *archivo*. *modo* debe ser el mismo modo usado para abrir *archivo*, o de otra forma pueden presentarse errores extraños cuando intente escribir y el modo de la secuencia sea diferente al modo sobre el archivo.

php_stream_fopen_tmpfile

(no version information, might be only in CVS)

php_stream_fopen_tmpfile -- Abrir un FILE* con tmpfile() y convertirlo en una secuencia

Descripción

php_stream * **php_stream_fopen_tmpfile** (void)

php_stream_fopen_tmpfile() devuelve una secuencia basada en un archivo temporal abierto con el modo "w+b". El archivo temporal será eliminado automáticamente cuando la secuencia es cerrada o el proceso es terminado.

php_stream_fopen_temporary_file

(no version information, might be only in CVS)

php_stream_fopen_temporary_file -- Generar un nombre de archivo temporal y abrir una secuencia sobre él

Descripción

php_stream * **php_stream_fopen_temporary_file** (const char * dir, const char * pref, char ** abierto)

php_stream_fopen_temporary_file() genera un nombre de archivo temporal en el directorio especificado por *dir* y con el prefijo *pref*. El nombre de archivo generado es devuelto en el parámetro *abierto*, el cual usted debe limpiar usando **efree()**. Una secuencia es abierta sobre ese nombre de archivo generado en modo "w+b". El archivo NO es eliminado automáticamente; usted es responsable de retirar el enlace o mover el archivo cuando haya terminado de trabajar con él.

Referencia de API de Sockets de Secuencia

Tabla de contenidos

php_stream_sock_open_from_socket

(no version information, might be only in CVS)

php_stream_sock_open_from_socket -- Convertir un descriptor de socket en una secuencia

Descripción

php_stream * **php_stream_sock_open_from_socket** (int socket, int persistente)

php_stream_sock_open_from_socket() devuelve una secuencia basada en el *socket*. *persistente* es una bandera que controla si la secuencia es abierta como una secuencia persistente. Por lo general, este parámetro será 0 en un gran número de casos.

php_stream_sock_open_host

(no version information, might be only in CVS)

php_stream_sock_open_host -- Abrir una conexión con un servidor huésped y devolver una secuencia

Descripción

php_stream * **php_stream_sock_open_host** (const char * host, unsigned short puerto, int tipo_sock, struct timeval * tiempo_espera, int persistente)

php_stream_sock_open_host() establece una conexión con el *host* y *puerto* especificados. *tipo_sock* especifica la semántica de conexión que debe aplicarse a la misma. Los valores de *tipo_sock* dependen del sistema, pero usualmente incluyen (como mínimo) **`SOCK_STREAM`** para secuencias basadas en conexiones secuenciales, confiables y de doble vía (TCP), o **`SOCK_DGRAM`** para mensajes sin conexión, no confiables y una longitud máxima (UDP).

persistente es una bandera que controla si la secuencia es abierta como una secuencia persistente. Por lo general, este parámetro será 0.

Si es diferente a NULL, *tiempo_espera* especifica un tiempo máximo para permitir que se establezca la conexión. Si el intento de conexión toma más tiempo que el valor de tiempo de espera, el intento de conexión es abortado y se devuelve NULL para indicar que la secuencia no pudo ser abierta.

> **Nota:** El valor de tiempo de espera no incluye el tiempo que toma realizar una consulta de DNS. La razón de esto es que no hay una forma portable de implementar una consulta DNS que no bloquee.
>
> El tiempo de espera sólo se aplica a la fase de conexión; si necesita establecer los tiempos de espera para operaciones posteriores de lectura o escritura, debería usar **php_stream_sock_set_timeout()** para configurar la duración de tiempo de espera para su secuencia una vez ésta ha sido abierta.

La API de secuencias no coloca restricciones sobre los valores que puede usar para *tipo_socket*, pero es recomendable que considere la portabilidad de los valores que elige antes de que libere su extensión.

php_stream_sock_open_unix

(no version information, might be only in CVS)

php_stream_sock_open_unix -- Abrir un socket de dominio Unix y convertirlo en una secuencia

Descripción

php_stream * **php_stream_sock_open_unix** (const char * ruta, int long_ruta, int persistente, struct timeval * tiempo_espera)

php_stream_sock_open_unix() intenta abrir el socket de dominio Unix especificado por *ruta*. *long_ruta* especifica la longitud de *ruta*. Si *tiempo_espera* es diferente a NULL, éste valor indica un periodo de tiempo máximo de espera para el intento de conexión. *persistente* indica si la secuencia debería ser abierta como una secuencia persistente. Por lo general, éste parámetro será 0 en la mayoría de casos.

> **Nota:** Esta función no funcionará bajo Windows, en donde no se implementan sockets de dominio Unix. Una posible excepción a esta regla ocurre si su binario de PHP fue compilado usando cygwin. Es recomendable que considere este aspecto de la portabilidad de su extensión antes de su lanzamiento.
>
> **Nota:** Esta función trata *ruta* en una manera segura con material binario, que es apropiada en sistemas con un espacio de nombres abstracto (como Linux), en donde el primer caracter de una ruta es un caracter NUL.

Estructuras de Secuencia

Tabla de contenidos

struct php_stream_wrapper -- Contiene propiedades de envolturas y apuntadores a operaciones
struct php_stream_wrapper_ops -- Contiene funciones miembro para una implementación de envoltura de secuencia
struct php_stream_filter -- Contiene propiedades de filtro y apuntadores a operaciones
struct php_stream_filter_ops -- Contiene funciones miembro para una implementación de filtro de secuencia

struct php_stream_statbuf

struct php_stream_statbuf -- Contiene información sobre un archivo o URL

Descripción

```
php_stream_statbuf
     struct statsb
```

`sb` es una estructura stat, normal, definida por el sistema.

struct php_stream_dirent

struct php_stream_dirent -- Contiene información sobre un archivo único durante el estudio de un directorio

Descripción

```
php_stream_dirent
     char d_name[MAXPATHLEN]
```

`d_name` contiene el nombre del archivo, relativo al directorio que está siendo estudiado.

struct php_stream_ops

struct php_stream_ops -- Contiene funciones miembro para una implementación de secuencia

Descripción

```
typedef struct _php_stream_ops {
             /* todas las secuencias DEBEN implementar estas operaciones */
             size_t (*write)(php_stream *stream, const char *buf, size_t count TSRMLS_D
             size_t (*read)(php_stream *stream, char *buf, size_t count TSRMLS_DC);
             int (*close)(php_stream *stream, int close_handle TSRMLS_DC);
             int (*flush)(php_stream *stream TSRMLS_DC);

             const char *label; /* nombre que describe esta clase de secuencia */

             /* estas operaciones son opcionales, y pueden definirse a NULL si la secue
              * soporta una operacion en particular */
            int (*seek)(php_stream *stream, off_t offset, int whence TSRMLS_DC);
            char *(*gets)(php_stream *stream, char *buf, size_t size TSRMLS_DC);
            int (*cast)(php_stream *stream, int castas, void **ret TSRMLS_DC);
            int (*stat)(php_stream *stream, php_stream_statbuf *ssb TSRMLS_DC);
        } php_stream_ops;
```

struct php_stream_wrapper

struct php_stream_wrapper -- Contiene propiedades de envolturas y apuntadores a operaciones

Descripción

```
struct _php_stream_wrapper  {
            php_stream_wrapper_ops *wops;   /* operaciones que puede realizar la envolt
            void *abstract;                 /* contexto para la envoltura */
            int is_url;                     /* de modo que PG(allow_url_fopen) pueda re

            /* soporte para que las envolturas devuelvas (multiples) mensajes de error
            int err_count;
            char **err_stack;
        } php_stream_wrapper;
```

struct php_stream_wrapper_ops

struct php_stream_wrapper_ops -- Contiene funciones miembro para una implementación de envoltura de secuencia

Descripción

```
typedef struct _php_stream_wrapper_ops {
            /* abrir/crear una secuencia envuelta */
            php_stream *(*stream_opener)(php_stream_wrapper *wrapper, char *filename, c
                    int options, char **opened_path, php_stream_context *context STREAM
            /* cerrar/destruir una secuencia envuelta */
            int (*stream_closer)(php_stream_wrapper *wrapper, php_stream *stream TSRMLS
            /* operacion stat sobre una secuencia envuelta */
            int (*stream_stat)(php_stream_wrapper *wrapper, php_stream *stream, php_str
            /* operacion stat sobre una URL */
            int (*url_stat)(php_stream_wrapper *wrapper, char *url, php_stream_statbuf
            /* abrir una secuencia de "directorio" */
            php_stream *(*dir_opener)(php_stream_wrapper *wrapper, char *filename, char
                    int options, char **opened_path, php_stream_context *context STREAM

            const char *label;

            /* Eliminar/Retirar Enlace de un archivo */
            int (*unlink)(php_stream_wrapper *wrapper, char *url, int options, php_stre
        } php_stream_wrapper_ops;
```

struct php_stream_filter

struct php_stream_filter -- Contiene propiedades de filtro y apuntadores a operaciones

Descripción

```
struct _php_stream_filter {
            php_stream_filter_ops *fops;
            void *abstract; /* para su uso por la implementacion de filtro */
            php_stream_filter *next;
            php_stream_filter *prev;
            int is_persistent;

            /* enlace a la secuencia y la cadena */
            php_stream_filter_chain *chain;

            /* paquetes en bufer */
            php_stream_bucket_brigade buffer;
        } php_stream_filter;
```

struct php_stream_filter_ops

struct php_stream_filter_ops -- Contiene funciones miembro para una implementación de filtro de secuencia

Descripción

```
typedef struct _php_stream_filter_ops {
            php_stream_filter_status_t (*filter)(
                    php_stream *stream,
                    php_stream_filter *thisfilter,
                    php_stream_bucket_brigade *buckets_in,
                    php_stream_bucket_brigade *buckets_out,
                    size_t *bytes_consumed,
                    int flags
                    TSRMLS_DC);

            void (*dtor)(php_stream_filter *thisfilter TSRMLS_DC);

            const char *label;
} php_stream_filter_ops;
```

Constantes de Secuencias

Tabla de contenidos

Opciones de apertura de secuencia

Opciones de apertura de secuencia -- Afecta la operación de las funciones de fábrica de secuencias

Descripción

Uno o más de éstos valores pueden ser combinados usando el operador OR.

`IGNORE_PATH`

Esta es la opción predeterminada para las secuencias; solicita que include_path no sea usado en busca del archivo requerido.

USE_PATH

Solicita que include_path sea usado en busca del archivo requerido.

IGNORE_URL

Solicita que las envolturas de URL registradas sean ignoradas cuando se abra la secuencia. Otras envolturas que no sean de URL serán tomadas en cuenta cuando se decodifique la ruta. No hay un valor opuesto para ésta bandera; la API de secuencias usa todas las envolturas registradas por defecto.

IGNORE_URL_WIN

En sistemas windows, éste es el ñalente a IGNORE_URL. En todos los otros sistemas, ésta bandera no tiene efecto alguno.

ENFORCE_SAFE_MODE

Solicita que la implementación interna de secuencia realice chequeos de safe_mode sobre el archivo antes de abrirlo. Al omitir ésta bandera se evitarán los chequeos de safe_mode y permite abrir cualquier archivo sobre el que el proceso de PHP tengo privilegios de acceso.

REPORT_ERRORS

Si ésta bandera se encuentra activa, y hubo un error durante la apertura del archivo o URL, la API de secuencias llamará la función php_error por usted. Esto es útil ya que la ruta puede contener información sobre nombres de usuario/contraseñas que no debería ser desplegada en la salida del navegador (cosa que sería un riesgo de seguridad). Cuando la API de secuencias genera un error, primero retira cualquier información de nombres de usuario/contraseñas de la ruta, haciendo que sea seguro mostrar el mensaje de error en el navegador.

STREAM_MUST_SEEK

Esta bandera es útil cuando su extensión realmente debe ser capaz de hacer búsquedas aleatorias en una secuencia. Algunas secuencias pueden no ser reubicables en su forma nativa, así que ésta bandera le pide a la API de secuencias que chequee si la secuencia soporta la reubicación. Si no lo hace, copiará la secuencia en un almacenamiento temporal (que puede tratarse de un archivo temporal o una secuencia en memoria) que sí soporte búsquedas. Por favor note que ésta bandera no es útil cuando usted quiere hacer búsquedas en la secuencia y escribir sobre ella, ya que la secuencia con la que trabaja puede no estar asociada con el recurso real que usted solicitó.

> **Nota:** Si el recurso solicitado es basado en red, ésta bandera causará que el proceso de apertura bloquee hasta que el contenido completo haya sido descargado.

STREAM_WILL_CAST

Si su extensión usa una biblioteca externa que espera un descriptor de archivo FILE*, puede usar esta bandera para solicitar a la API de secuencias que abra el recurso pero que evite el uso de búferes. Puede usar entonces **php_stream_cast()** para recuperar el apuntador FILE* o descriptor de archivo que la biblioteca requiere.

Esto es particularmente útil cuando accede a URLs HTTP en donde el inicio de los datos

reales de la secuencia se encuentra después de un desplazamiento indeterminado al interior de la secuencia.

Ya que esta opción deshabilita el uso de búferes en el nivel del API de secuencias, usted puede experimentar un rendimiento inferior cuando usa funciones de secuencias sobre ella; esto es considerado aceptable ya que usted le ha indicado a la API que estará usando las funciones para adaptar la implementación interna de secuencias. Use esta opción únicamente cuando esté seguro de que la necesita.

IX. FAQ: Preguntas frecuentes

Tabla de contenidos

Capítulo 64. General Information

This section holds the most general questions about PHP: what it is and what it does.

1. What is PHP?

From the preface of the manual:

PHP is an HTML-embedded scripting language. Much of its syntax is borrowed from C, Java and Perl with a couple of unique PHP-specific features thrown in. The goal of the language is to allow web developers to write dynamically generated pages quickly.

A nice introduction to PHP by Stig SÃƒÂ¦ther Bakken can be found here on the Zend website. Also, much of the PHP Conference Material is freely available.

2. What does PHP stand for?

PHP stands for *PHP: Hypertext Preprocessor*. This confuses many people because the first word of the acronym is the acronym. This type of acronym is called a recursive acronym. The curious can visit Free On-Line Dictionary of Computing for more information on recursive acronyms.

3. What is the relation between the versions?

PHP/FI 2.0 is an early and no longer supported version of PHP. PHP 3 is the successor to PHP/FI 2.0 and is a lot nicer. PHP 4 is the current generation of PHP, which uses the Zend engine under the hood. PHP 5 uses Zend engine 2 which, among other things, offers many additional OOP features. PHP 5 is experimental.

4. Can I run several versions of PHP at the same time?

Yes. See the `INSTALL` file that is included in the PHP 4 source distribution. Also, read the related appendix.

5. What are the differences between PHP 3 and PHP 4?

There are a couple of articles written on this by the authors of PHP 4. Here's a list of some of the more important new features:

- Extended API module
- Generalized build process under UNIX
- Generic web server interface that also supports multi-threaded web servers
- Improved syntax highlighter
- Native HTTP session support
- Output buffering support
- More powerful configuration system
- Reference counting

Please see the What's new in PHP 4 overview for a detailed explanation of these features and more. If you're migrating from PHP 3 to PHP 4, also read the related appendix.

6. I think I found a bug! Who should I tell?

You should go to the PHP Bug Database and make sure the bug isn't a known bug. If you don't see it in the database, use the reporting form to report the bug. It is important to use the bug database instead of just sending an email to one of the mailing lists because the bug will have a tracking number assigned and it will then be possible for you to go back later and check on the status of the bug. The bug database can be found at http://bugs.php.net/.

Capítulo 65. Listas de correo

Esta sección contiene preguntas sobre cómo ponerse en contacto con la comunidad de PHP. La mejor manera es a través de las listas de correo.

1. ¿Existen listas de correo sobre PHP?

¡Por supuesto! Existen varias listas de correo para varios asuntos. Una lista completa de listas de correo puede encontrarse en nuestra página de Soporte.

La lista de correo más general es *php-general*. Para suscribirse, envíe un correo a php-general-subscribe@lists.php.net. No necesita incluir nada especial en el asunto o cuerpo del mensaje. Para cancelar la suscripción, envíe un correo a php-general-unsubscribe@lists.php.net.

También puede iniciar y cancelar suscripciones usando la interfaz web en nuestra página de Soporte.

2. ¿Existen otras comunidades?

Existe un sinnúmero de ellas alrededor del mundo. Nosotros contamos con enlaces, por ejemplo, hacia servidores IRC y listas de correo en diferentes lenguajes en nuestra página de Soporte.

3. ¡Ayuda! ¡Aparentemente no puedo iniciar/cancelar suscripciones en una de las listas de correo!

Si tiene problemas con la suscripción o cancelación de una suscripción en la lista de correo php-general, puede ser debido a que el software de la lista de correo no puede adivinar la dirección de correo apropiada para usar. Si su dirección de correo electrónico era *joeblow@example.com*, puede enviar su petición de suscripción a *php-general-subscribe-joeblow=example.com@lists.php.net*, o su petición de cancelación del servicio a *php-general-unsubscribe-joeblow=example.com@lists.php.net*. Use direcciones similares para las otras listas de correo.

4. ¿Existe un archivo de las listas de correo en alguna parte?

Si, encontrará una lista de sitios que mantienen archivos en la página de Soporte. Los artículos de las listas de correo también son archivados como mensajes de noticias. Puede acceder al servidor de noticias en news://news.php.net/ con un cliente de este servicio. También existe una interfaz web experimental para el servidor de noticias en http://news.php.net/

5. ¿Qué puedo preguntar a la lista de correo?

Dado que PHP está creciendo más y más cada día, el tráfico en la lista de correo php-general se ha incrementado y en el momento ronda los 150 a 200 mensajes por día. Debido a esto, es del interés de todos que use la lista como un último recurso una vez ha probado todos los otros caminos.

Antes de enviar un mensaje a la lista, por favor eche un vistazo a este FAQ y el manual para ver si puede encontrar ayuda en tales sitios. Si no encuentra nada, pruebe con los archivos de la lista de correo (vea la mención de este tema en un punto anterior). Si tiene problemas con la instalación o configuración de PHP, por favor lea en su totalidad la documentación incluida y los documentos README. Si aun no puede encontrar información que le ayude, entonces es más que bienvenido a usar la lista de correo.

Antes de formular preguntas, puede que quiera leer el documento sobre Cómo Formular Preguntas de Forma Inteligente, cosa que es una buena idea para todos.

6. ¿Qué información debo incluir cuando envío mensajes a la lista de correo?

Mensajes del tipo "¡No logro que PHP funcione! ¡Ayúdenme! ¿Qué está pasando?" son absolutamente inútiles para cualquier persona. Si está sufriendo dificultades con la configuración y ejecución de PHP, debe incluir información sobre el sistema operativo que usa, la versión de PHP que está tratando de instalar, en qué forma adquirió el software (pre-compilado, CVS, RPMs, etc), qué ha hecho hasta el momento, en qué punto se interrumpió el proceso, y el mensaje de error exacto.

Esto se aplica también para cualquier otro problema. Debe incluir información sobre lo que ha hecho, en dónde quedó bloqueado, qué está tratando de hacer y, si se aplica, los mensajes de error exactos. Si tiene problemas con su código fuente, necesita incluir la parte de éste que no está trabajando. ¡Aunque, no incluya más código del necesario! Hace que el mensaje sea difícil de leer, y muchas personas pueden simplemente ignorarlo por esta razón. Si no está seguro sobre la cantidad de información a incluir en el correo, es mejor que incluya de más.

Otra cosa importante a tener en cuenta es ofrecer un resumen de su problema en la línea de asunto. Un asunto como "¡AYÃƒÅ¡DENME!!!" o "¿Cuál es el problema aquí?" será ignorado por la mayoría de lectores.

Y, por último, es recomendable que lea el documento sobre Cómo Formular Preguntas de Forma Inteligente, cosa que será de ayuda para todos, especialmente para usted.

Capítulo 66. Obtención de PHP

Esta sección contiene detalles sobre las ubicaciones de descarga de PHP, y asuntos particulares a ciertos sistemas operativos.

1. ¿Desde dónde puedo obtener PHP?

Puede descargar PHP desde cualquiera de los miembros de la red de sitios PHP. Estos pueden encontrarse en http://www.php.net/. También puede usar CVS anónimo para obtener la última versión del código fuente. Para más información, visite http://www.php.net/anoncvs.php.

2. ¿Hay versiones binarias pre-compiladas disponibles?

Nosotros sólo distribuimos binarios pre-compilados para sistemas Windows, dado que no tenemos la capacidad de compilar PHP para cada plataforma Linux/Unix popular con todas las combinaciones de extensiones. También note que muchas distribuciones Linux vienen con PHP integrado en la actualidad. Los binarios de Windows pueden descargarse desde nuestra página de

Descargas, para encontrar binarios Linux, por favor visite el sitio web de su distribución.

3. ¿En dónde puedo encontrar las bibliotecas requeridas para compilar algunas de las extensiones opcionales de PHP?

> **Nota:** Aquellas marcadas con el signo * son líbrerias que no son seguras en entornos multi-hilos, y no deberían ser usadas con PHP como módulo de servidor en los servidores web milti-hilos de Windows (IIS, Netscape). Por el momento, esto no tiene importancia en los entornos Unix.

- LDAP (Unix).
- LDAP* (Unix).
- LDAP (Unix/Win) : Netscape Directory (LDAP) SDK 1.1.
- servidor LDAP gratuito.
- Berkeley DB2 (Unix/Win) : http://www.sleepycat.com/.
- SNMP* (Unix): .
- GD* (Unix/Win).
- mSQL* (Unix).
- PostgreSQL (Unix).
- IMAP* (Win/Unix).
- Sybase-CT* (Linux, libc5) : Disponible localmente.
- FreeType (libttf):.
- ZLib (Unix/Win32).
- procesador XML expat (Unix/Win32).
- PDFLib.
- mcrypt.
- mhash.
- t1lib.
- dmalloc.
- aspell.
- readline.

4. ¿Cómo logro que éstas bibliotecas funcionen?

Necesitará seguir las instrucciones provistas con la biblioteca. Algunas de estas bibliotecas son detectadas automáticamente cuando ejecuta el script 'configure' de PHP (como la biblioteca GD), y otras tendrán que ser habilitadas usando opciones '*--with-EXTENSION*' con '*configure*'. Ejecute '*configure --help*' para recibir un listado de éstas opciones.

5. He conseguido la última versión del código PHP desde el repositorio CVS en mi máquina Windows, ¿qué necesito para su compilación?

Primero, necesita Microsoft Visual C++ v6 (la versión 5 puede que funcione, pero nosotros usamos la versión 6), y necesitará algunos archivos de soporte. Vea la sección del manual sobre la compilación de PHP desde las fuentes en Windows.

6. ¿En dónde encuentro el Archivo de Capacidades del Navegador?

Puede encontrar un archivo `browscap.ini` en http://www.garykeith.com/browsers/downloads.asp.

Capítulo 67. Database issues

This section holds common questions about relation between PHP and databases. Yes, PHP can access virtually any database available today.

1. I heard it's possible to access Microsoft SQL Server from PHP. How?

On Windows machines, you can simply use the included ODBC support and the correct ODBC driver.

On Unix machines, you can use the Sybase-CT driver to access Microsoft SQL Servers because they are (at least mostly) protocol-compatible. Sybase has made a free version of the necessary libraries for Linux systems. For other Unix operating systems, you need to contact Sybase for the correct libraries. Also see the answer to the next question.

2. Can I access Microsoft Access databases?

Yes. You already have all the tools you need if you are running entirely under Windows 9x/Me, or NT/2000, where you can use ODBC and Microsoft's ODBC drivers for Microsoft Access databases.

If you are running PHP on a Unix box and want to talk to MS Access on a Windows box you will need Unix ODBC drivers. OpenLink Software has Unix-based ODBC drivers that can do this. There is a free pilot program where you can download an evaluation copy that doesn't expire and prices start at $675 for the commercial supported version.

Another alternative is to use an SQL server that has Windows ODBC drivers and use that to store the data, which you can then access from Microsoft Access (using ODBC) and PHP (using the built in drivers), or to use an intermediary file format that Access and PHP both understand, such as flat files or dBase databases. On this point Tim Hayes from OpenLink software writes:

```
Using another database as an intermediary is not a good idea, when you can
      use ODBC from PHP straight to your database - i.e. with OpenLink's drivers. If
      you do need to use an intermediary file format, OpenLink have now released
      Virtuoso (a virtual database engine) for NT, Linux and other unix platforms.
      Please visit our website for a free download.
```

One option that has proven successful is to use MySQL and its MyODBC drivers on Windows and synchronizing the databases. Steve Lawrence writes:

- Install MySQL on your platform according to instructions with MySQL. Latest available from www.mysql.com (get it from your mirror!). No special configuration required except when you set up a database, and configure the user account, you should put % in the host field, or the host name of the Windows computer you wish to access MySQL with. Make a note of your server name, username, and password.

- Download the MyODBC for Windows driver from the MySQL site. Latest release is myodbc-2_50_19-win95.zip (NT available too, as well as source code). Install it on your Windows machine. You can test the operation with the utilities included with this program.

- Create a user or system dsn in your ODBC administrator, located in the control panel. Make up a dsn name, enter your hostname, user name, password, port, etc for you MySQL database configured in step 1.

- Install Access with a full install, this makes sure you get the proper add-ins.. at the least you will need ODBC support and the linked table manager.

- Now the fun part! Create a new access database. In the table window right click and select Link Tables, or under the file menu option, select Get External Data and then Link Tables. When the file browser box comes up, select files of type: ODBC. Select System dsn and the name of your dsn created in step 3. Select the table to link, press OK, and presto! You can now open the table and add/delete/edit data on your MySQL server! You can also build queries, import/export tables to MySQL, build forms and reports, etc.

Tips and Tricks:

- You can construct your tables in Access and export them to MySQL, then link them back in. That makes table creation quick.

- When creating tables in Access, you must have a primary key defined in order to have write access to the table in access. Make sure you create a primary key in MySQL before linking in access

- If you change a table in MySQL, you have to re-link it in Access. Go to tools>add-ins>linked table manager, cruise to your ODBC DSN, and select the table to re-link from there. you can also move your dsn source around there, just hit the always prompt for new location checkbox before pressing OK.

3. I upgraded to PHP 4, and now mysql keeps telling me "Warning: MySQL: Unable to save result set in ...". What's up?

Most likely what has happened is, PHP 4 was compiled with the '--with-mysql' option, without specifying the path to MySQL. This means PHP is using its built-in MySQL client library. If your

system is running applications, such as PHP 3 as a concurrent Apache module, or auth-mysql, that use other versions of MySQL clients, then there is a conflict between the two differing versions of those clients.

Recompiling PHP 4, and adding the path to MySQL to the flag, '--with-mysql=/your/path/to/mysql' usually solves the problem.

4. After installing shared MySQL support, Apache dumps core as soon as libphp4.so is loaded. Can this be fixed?

If your MySQL libs are linked against pthreads this will happen. Check using ldd. If they are, grab the MySQL tarball and compile from source, or recompile from the source rpm and remove the switch in the spec file that turns on the threaded client code. Either of these suggestions will fix this. Then recompile PHP with the new MySQL libs.

5. Why do I get an error that looks something like this: "Warning: 0 is not a MySQL result index in <file> on line <x>" or "Warning: Supplied argument is not a valid MySQL result resource in <file> on line <x>?

You are trying to use a result identifier that is 0. The 0 indicates that your query failed for some reason. You need to check for errors after submitting a query and before you attempt to use the returned result identifier. The proper way to do this is with code similar to the following:

```
$result = mysql_query("SELECT * FROM tables_priv");
if (!$result) {
    echo mysql_error();
    exit;
}
```

or

```
$result = mysql_query("SELECT * FROM tables_priv")
    or die("Bad query: ".mysql_error());
```

Capítulo 68. Instalación

Esta sección contiene preguntas comunes sobre el modo de instalar PHP. PHP se encuentra disponible para casi cualquier SO (excepto quizás por MacOS antes de OSX), y casi cualquier servidor web.

Para instalar PHP, siga las instrucciones del archivo INSTALL ubicado en la distribución. Los usuarios de windows también deberían leer el archivo install.txt. También existen algunas pistas útiles para usuarios de windows aquí.

8. Unix/Windows: He instalado PHP, pero cuando intento acceder a un script PHP a través de mi navegador, recibo un error 500 de servidor.
9. Algunos sistemas operativos: He instalado PHP sin errores, pero cuando intento iniciar apache, recibo errores de símbolos indefinidos:

```
[mi_maquina:usuario /src/php4] root# apachectl configtest
 apachectl: /usr/local/apache/bin/httpd Undefined symbols:
  _compress
  _uncompress
```

10. Windows: He instalado PHP, pero cuando intento acceder a un script PHP a través de mi navegador, recibo el error:

```
cgi error:
 The specified CGI application misbehaved by not
 returning a complete set of HTTP headers.
 The headers it did return are:
```

11. Windows: He seguido todas las instrucciones, ¡pero aun no logro que PHP e IIS trabajen juntos!
12. Cuando corro PHP como CGI con IIS, PWS, OmniHTTPD o Xitami, recibo el siguiente error: *Security Alert! PHP CGI cannot be accessed directly.*.
13. ¿Cómo se si mi `php.ini` está siendo encontrado y leído? Pareciera que mis cambios no están siendo implementados.
14. ¿Cómo agrego mi directorio de PHP a la variable *PATH* en Windows?
15. ¿Cómo hago que el archivo `php.ini` se encuentre disponible para PHP en windows?

1. ¿Porqué no debo usar Apache 2 en un entorno en producción?

La siguiente respuesta se encuentra basada en este extracto modificado de un correo por Rasmus Lerdorf.

Apache 2 es una versión escrita desde ceros y una completa modificación de arquitectura a partir de Apache 1. No es como ir de PHP 3 a PHP4, o de PHP 4 a PHP 5. Hay bastante código que es común, y ciertamente la arquitectura base de PHP no ha cambiado a través de los años. Asi que comparar Apache 1 y Apache 2 contra PHP 4 y PHP 5 no tiene sentido. La arquitectura ha sido probada a lo largo de los años y el código, aunque es un poco inmanejable en algunas partes, es una entidad conocida. PHP fue diseñado desde los primeros días contra aquella arquitectura básica de Apache 1 y funciona extremadamente bien cuando se usa con éste.

La característica más importante que atrae a la gente a Apache 2 es el uso de hilos. En Windows, en donde la mayoría de bibliotecas básicos son, y deben ser, seguras con hilos, Apache 2 realmente tiene sentido y sería bueno resolver los pequeños problemas en esa plataforma. Sin embargo, en UNIX existen bastante bibliotecas básicas en donde no se conoce su seguridad con hilos. Y no se trata de extensiones PHP, se trata de bibliotecas de terceros por debajo de los cientos de extensiones de PHP. Es realmente difícil determinar si cierta biblioteca externa es segura con hilos. Existen bastantes variables involucradas, incluyendo el SO, la versión del SO, la biblioteca libc, la versión de esa biblioteca e incluso, en algunas plataformas, las banderas de compilación usadas para generar las bibliotecas. Y para hacerlo aun más divertido, rastrear un problema de seguridad con hilos se encuentra bastante cerca a lo imposible. Cientos de personas bien pueden declarar que Apache+PHP+cualquier/extensión funciona perfectamente para ellos, pero quizás ellos sólo reciben un millón de peticiones al día. Luego llega otro usuario quien recibe 100 millones de peticiones al día y usa una avanzada máquina con doble-procesador y todo se arruina ya que de repente ahora la oportunidad para alguna pequeña condición de carrera se ha hecho mucho mayor debido a las velocidades más rápidas de cpu, el segundo cpu y la mayor frecuencia de las peticiones. Y el reporte del fallo que recibimos de este usuario será algo similar a:

> A veces no funciona. La mayoría de veces funciona bien, pero de repente de vez en cuando no. El error es diferente cada vez y no tengo idea de cómo reproducirlo, ¡pero arréglenlo de inmediato!!!

¿Qué podemos hacer al respecto?

Hay un número de razones técnicas (que pueden resolverse) por las que Rasmus no piensa que Apache2+PHP sea una buena idea en un entorno en producción, pero dejándolas de lado, realmente todo converge a un simple concepto:

PHP es pegamento. Es el pegamento usado para construir interesantes aplicaciones web, uniendo docenas de bibliotecas externas y haciéndolas parecer una entidad coherente a través de una interfaz de lenguaje intuitiva y fácil de aprender. La flexibilidad y poder de PHP depende de la estabilidad y robustez de la plataforma subyacente. Necesita de un SO que funcione, un servidor web que funcione y unas bibliotecas externas que funcionen para unirlo todo. Cuando uno cualquiera de estos elementos deja de trabajar, PHP necesita identificar los problemas y solucionarlos rápidamente. Al hacer más complejo el marco de referencia base, no disponiendo de hilos de ejecución completamente separados, segmentos de memoria completamente separados y una caja de arena segura para que cada petición trabaje, se introduce una base débil al sistema de PHP.

Usar la versión mpm anterior a la bifurcación ocurrida en Apache 2 para evitar el uso de hilos es posible, y sí, también funciona usar un mecanismo independiente fastcgi para evitar los hilos, pero entonces se estarían evitando características que definen al servidor web. En este punto de desarrollo, Rasmus aun afirma que se está mejor siguiendo con Apache 1 para servir páginas PHP, con la nota de precaución de que Apache 1 opera bastante mal en Windows.

2. Unix/Windows: ¿En dónde debe estar ubicado mi archivo `php.ini`?

Por defecto, en Unix debe estar en `/usr/local/lib`, lo que es, `<ruta-instalacion>/lib`. La mayoría de personas querrán modificar éste valor en tiempo de compilación con la bandera --with-config-file-path. Podría, por ejemplo, definirla como:

```
--with-config-file-path=/etc
```

Y luego copiaría `php.ini-dist` desde su distribución a `/etc/php.ini` y lo editaría para hacer cualquier modificación local que desee.

```
--with-config-file-scan-dir=PATH
```

En windows, la ruta predeterminada para el archivo `php.ini` es el directorio de windows. Si está usando el servidor web Apache, `php.ini` será buscado primero en el directorio de instalación de Apache, p.ej. `c:\program files\apache group\apache`. De este modo, puede contar con diferentes archivos `php.ini` para diferentes versiones de Apache en la misma máquina.

Vea también el capítulo sobre el archivo de configuración.

3. Unix: Instalé PHP, ¡pero cada vez que cargo un documento, recibo el mensaje 'Document Contains No Data'! ¿Qué está pasando aquí?

Esto probablemente quiere decir que PHP está sufriendo algún tipo de dificultad y está produciendo volcados de memoria. Eche un vistazo a su registro de errores del servidor para ver si éste es el caso, y luego trate de reproducir el problema con un pequeño caso de prueba. Si sabe cómo usar 'gdb', es muy útil cuando puede proveer un backtrace con su reporte de fallo para ayudar a los desarrolladores a ubicar el problema. Si está usando PHP como módulo de Apache, intente algo como:

- Detener todos sus procesos httpd
- gdb httpd

- Detener todos sus procesos httpd
- > run -X -f /ruta/hacia/httpd.conf
- Luego recuperar la URL que causa el problema con su navegador
- > run -X -f /ruta/hacia/httpd.conf
- Si está recibiendo un volcado de memoria, gdb debe informarle de ésto.
- escriba: bt
- Ahora debe incluir su backtrace en su reporte de fallo. Éste debe ser enviado desde http://bugs.php.net/

Si su script usa las funciones de expresiones regulares (**ereg()** y amigas), debe asegurarse de que compiló PHP y Apache con el mismo paquete de expresiones regulares. Esto debe pasar automáticamente con PHP y Apache 1.3.x

4. Unix: Instalé PHP usando RPMS, ¡pero Apache no está procesando las páginas PHP¡ ¿Qué está pasando aquí?

Asumiendo que ha instalado tanto Apache como PHP desde paquetes RPM, necesita remover los caracteres de comentario, o agregar algunas o todas de las siguientes líneas en su archivo `httpd.conf`:

```
# Modulos Extra
AddModule mod_php.c
AddModule mod_php3.c
AddModule mod_perl.c

# Modulos Extra
LoadModule php_module         modules/mod_php.so
LoadModule php3_module        modules/libphp3.so     # para PHP 3
LoadModule php4_module        modules/libphp4.so     # para PHP 4
LoadModule perl_module        modules/libperl.so
```

Y agregar:

```
AddType application/x-httpd-php3 .php3    # para PHP 3
AddType application/x-httpd-php .php      # para PHP 4
```

... a las propiedades globales, o a las propiedades del VirtualDomain para el cual desea tener soporte de PHP.

5. Unix: Instalé PHP 3 usando RPMS, ¡pero no compila con el soporte de bases de datos que necesito! ¿Qué está pasando?

Debido a la forma en que se compila PHP 3, no es sencillo crear un RPM de PHP flexible completo. Este problema es atendido en PHP 4. Para PHP 3, actualmente le sugerimos que use el mecanismo descrito en el archivo INSTALL.REDHAT en la distribución de PHP. Si insiste en usar una versión RPM para PHP 3, continúe leyendo...

Los empaquetadores de RPM están configurando los RPMS para ser instalados sin soporte de bases de datos para simplificar las instalaciones *y* porque los RPMS usan /usr/ en lugar del directorio / usr/local/ estándar para los archivos. Usted necesita decirle al archivo spec del RPM cuáles bases de datos soportar y la ubicación del nivel más alto de su servidor de bases de datos.

Este ejemplo explicará el proceso de agregar soporte para el popular servidor de bases de datos MySQL, usando la instalación de módulo para Apache.

Por supuesto, toda esta información puede ser ajustada para cualquier servidor de bases de datos soportada por PHP. Asumiremos que ha instalado MySQL y Apache completamente con RPMS para este ejemplo también.

- Primero remueva mod_php3 :
  ```
  rpm -e mod_php3
  ```
- Luego obtenga el rpm fuente e INSTÃƒÂ LELO, NO use --rebuild
  ```
  rpm -Uvh mod_php3-3.0.5-2.src.rpm
  ```
- Luego edite el archivo `/usr/src/redhat/SPECS/mod_php3.spec`

  En la sección %build agrege el soporte de bases de datos que desea, y la ruta.

  Para MySQL, usted agregaría *--with-mysql=/usr* La sección %build se verá algo como:
  ```
  ./configure --prefix=/usr \
  --with-apxs=/usr/sbin/apxs \
  --with-config-file-path=/usr/lib \
  --enable-debug=no \
  --enable-safe-mode \
  --with-exec-dir=/usr/bin \
  --with-mysql=/usr \
  --with-system-regex
  ```
- Una vez está hecha esta modificación, compile el rpm binario de este modo:
  ```
  rpm -bb /usr/src/redhat/SPECS/mod_php3.spec
  ```
- Luego instale el rpm
  ```
  rpm -ivh /usr/src/redhat/RPMS/i386/mod_php3-3.0.5-2.i386.rpm
  ```

Asegúrese de reiniciar Apache, y ahora tendrá PHP 3 con soporte para MySQL usando RPM's. Note que probablemente es mucho más fácil simplemente compilar a partir del tarball de distribución de PHP 3 y seguir las instrucciones encontradas en `INSTALL.REDHAT`.

6. Unix: He aplicado el parche de extensiones FrontPage a Apache, y de pronto PHP dejó de funcionar. ¿Es PHP incompatible con las extensiones FrontPage de Apache?

No, PHP trabaja bien con las extensiones FrontPage. El problema es que el parche de FrontPage modifica varias estructuras de Apache en las que depende PHP. Recompilar PHP (usando 'make clean ; make') después de que se ha aplicado el parche FP debe solucionar el problema.

7. Unix/Windows: He instalado PHP, pero cuando intento acceder a un script PHP a través de mi navegador, recibo una pantalla en blanco.

Seleccione la acción 'ver código fuente' en el navegador web y probablemente encontrará el código fuente de su script PHP. Esto quiere decir que el servidor web no envió el script a PHP para su interpretación. Algo está mal en la configuración del servidor - revise cuidadosamente la configuración del servidor con las instrucciones de instalación de PHP.

8. Unix/Windows: He instalado PHP, pero cuando intento acceder a un script PHP a través de mi navegador, recibo un error 500 de servidor.

Algo falló cuando el servidor intentó ejecutar PHP. Para poder ver un mensaje de error más útil, desde la línea de comandos, vaya al directorio que contiene el ejecutable PHP (`php.exe` en Windows) y ejecute **php -i**. Si PHP tiene algún problema corriendo, entonces se desplegará un mensaje de error apropiado, el cual le dará una pista sobre lo que debe hacer a continuación. Si recibe una pantalla llena de códigos HTML (la salida de la función **phpinfo()**) entonces PHP está funcionando, y su problema puede estar relacionado con la configuración de su servidor, la cual debe revisar de nuevo.

9. Algunos sistemas operativos: He instalado PHP sin errores, pero cuando intento iniciar apache, recibo errores de símbolos indefinidos:

```
[mi_maquina:usuario /src/php4] root# apachectl configtest
 apachectl: /usr/local/apache/bin/httpd Undefined symbols:
  _compress
  _uncompress
```

Esto no tiene nada que ver con PHP en realidad, sino con las bibliotecas de cliente de MySQL. Algunas necesitan *--with-zlib*, otras no. Esto tema también se cubre en el FAQ sobre MySQL.

10. Windows: He instalado PHP, pero cuando intento acceder a un script PHP a través de mi navegador, recibo el error:

```
cgi error:
 The specified CGI application misbehaved by not
 returning a complete set of HTTP headers.
 The headers it did return are:
```

El mensaje de error significa que PHP falló en su intento de producir una salida. Para poder ver un mensaje de error más útil, desde la línea de comandos, vaya al directorio que contiene el ejecutable de PHP (`php.exe` en Windows) y ejecute **php -i**. Si PHP tiene algún problema ejecutándose, entonces se desplegará un mensaje de error apropiado, el cual le dará una pista sobre lo que necesita hacer a continuación. Si recibe una pantalla llena de códigos HTML (la salida de la función **phpinfo ()**) entonces PHP está funcionando.

Una vez PHP esté trabajando desde la línea de comandos, intente acceder al script desde el navegador nuevamente. Si aun falla, entonces la causa puede ser alguna de las siguientes:

- Los permisos de archivo en su script PHP, `php.exe`, `php4ts.dll`, `php.ini` o cualquier extensión PHP que esté tratando cargar son de tal forma que el usuario anónimo de internet *ISUR_<nombre_maquina>* no tiene acceso a éstos recursos.

- El archivo del script no existe (o posiblemente no se encuentra en donde cree que está, relativo a su directorio web raíz). Note que para IIS, puede atrapar este error seleccionando la caja 'check file exists' cuando esté configurando la gestión de scripts bajo Internet Services Manager. Si un archivo de script no existe, entonces el servidor devolverán un error 404 en su lugar. También existe el beneficio adicional de que IIS se encargará de cualquier autenticación requerida por usted, basado en los permisos NTLanMan en su archivo de script.

11. Windows: He seguido todas las instrucciones, ¡pero aun no logro que PHP e IIS trabajen juntos!

¡Asegúrese de que cualquier usuario que necesite ejecutar un script PHP tenga los permisos para ejecutar `php.exe`! IIS usa un usuario anónimo que es agregado al momento de instalar IIS. Este usuario necesita privilegios sobre `php.exe`. También, cualquier usuario autenticado necesitará permisos para ejecutar `php.exe`. Y para IIS4, necesita decirle que PHP es un motor de scripts. Asimismo, querrá leer este faq.

12. Cuando corro PHP como CGI con IIS, PWS, OmniHTTPD o Xitami, recibo el siguiente error: *Security Alert! PHP CGI cannot be accessed directly.*.

Debe definir la directiva cgi.force_redirect como *0*. Su valor predeterminado es *1*, así que asegúrese de que la directiva no se encuentre comentada (con un *;*). Como todas las directivas, ésta se define en `php.ini`

Debido a que su valor predeterminado es *1*, es importante que esté 100% seguro de que el archivo `php.ini` correcto está siendo leído. Lea este faq para más detalles.

13. ¿Cómo se si mi `php.ini` está siendo encontrado y leído? Pareciera que mis cambios no están siendo implementados.

Para asegurarse de que su `php.ini` está siendo leído por PHP, haga un llamado a **phpinfo()**, y cerca del comienzo encontrará un listado llamado *Configuration File (php.ini)*. Éste le dirá en dónde está buscando PHP el archivo `php.ini` y si está siendo leído o no. Si sólo existe un directorio PATH, entonces no está siendo leído y debe colocar su `php.ini` en ese directorio. Si `php.ini` es incluido con la ruta PATH, entonces está siendo leído.

Si `php.ini` está siendo leido y está ejecutando PHP como un módulo, entonces asegúrese de reiniciar su navegador web después de hacer cambios a `php.ini`

14. ¿Cómo agrego mi directorio de PHP a la variable *PATH* en Windows?

En Windows NT, 2000, XP y 2003:

- Diríjase al Panel de Control y abra el ícono de Sistema (Inicio -> Configuración -> Panel de Control -> Sistema, o simplemente Inicio -> Panel de Control -> Sistema en Windows XP/2003)
- Vaya a la solapa Avanzado
- Pulse el botón 'Variables de Entorno'
- Consulte el panel de 'Variables de Sistema'
- Encuentre la entrada Path (puede que tenga que desplazarse por la lista para encontrarla)
- Haga doble clic sobre la entrada Path
- Ingrese su directorio PHP al final, incluyendo ';' al comienzo (p.ej. *;C:\php*)
- Presione Aceptar y reinicie su máquina

En Windows 98/Me necesita editar el archivo `autoexec.bat`:

- Abra el Bloc de Notas (Inicio -> Ejecutar y escriba notepad)
- Abra el archivo `C:\autoexec.bat`
- Ubique la línea con PATH=C:\WINDOWS;C:\WINDOWS\COMMAND;..... y agregue: *;C:\php* al final de la línea
- Guarde el archivo y reinicie su máquina

El manual de PHP solía promover la copia de archivos al directorio system de Windows, esto es porque el directorio (`C:\Windows`, `C:\WINNT`, etc.) se encuentra por defecto en el valor PATH del sistema. Copiar archivos en el directorio system de Windows se considera obsoleto desde hace bastante tiempo, y puede causar problemas.

15. ¿Cómo hago que el archivo `php.ini` se encuentre disponible para PHP en windows?

Existen varias formas de lograr esto. Si está usando Apache, lea sus instrucciones específicas de instalación (Apache 1, Apache 2), o de lo contrario debe definir la variable de entorno *PHPRC*:

En Windows NT, 2000, XP y 2003:

- Diríjase al Panel de Control y abra el ícono de Sistema (Inicio -> Configuración -> Panel de Control -> Sistema, o simplemente Inicio -> Panel de Control -> Sistema para Windows XP/2003)
- Vaya a la solapa de Avanzado
- Pulse sobre el botón 'Variables de Entorno'
- Consulte el panel de 'Variables de sistema'
- Pulse sobre 'Nueva' e ingrese 'PHPRC' como el nombre de variable y el directorio en donde se encuentra `php.ini` como el valor de la variable (p.ej. *C:\php*)
- Pulse aceptar y reinicie su máquina

En Windows 98/Me necesita editar el archivo `autoexec.bat`:

- Abre el Bloc de Notas (Inicio -> Ejecutar e ingrese notepad)
- Abra el archivo `C:\autoexec.bat`
- Agrege una nueva línea al final del archivo: *set PHPRC C:\php* (reemplace *C:\php* con el directorio en donde se encuentra `php.ini`)
- Guarde el archivo y reinicie su máquina

Capítulo 69. Build Problems

This section gathers most common errors that occur at build time.

1. I got the latest version of PHP using the anonymous CVS service, but there's no configure script!

You have to have the GNU autoconf package installed so you can generate the configure script from `configure.in`. Just run **./buildconf** in the top-level directory after getting the sources from the CVS server. (Also, unless you run configure with the *--enable-maintainer-mode* option, the configure script will not automatically get rebuilt when the `configure.in` file is updated, so you should make sure to do that manually when you notice configure.in has changed. One symptom of this is finding things like @VARIABLE@ in your Makefile after configure or `config.status` is run.)

2. I'm having problems configuring PHP to work with Apache. It says it can't find `httpd.h`, but it's right where I said it is!

You need to tell the configure/setup script the location of the top-level of your Apache source tree. This means that you want to specify *--with-apache=/path/to/apache* and *not --with-apache=/path/to/apache/src*.

3. While configuring PHP (*./configure*), you come across an error similar to the following:

```
checking lex output file root... ./configure: lex: command not found
       configure: error: cannot find output from lex; giving up
```

Be sure to read the installation instructions carefully and note that you need both flex and bison installed to compile PHP. Depending on your setup you will install bison and flex from either source or a package, such as a RPM.

4. When I try to start Apache, I get the the following message:

```
fatal: relocation error: file /path/to/libphp4.so:
       symbol ap_block_alarms: referenced symbol not found
```

This error usually comes up when one compiles the Apache core program as a DSO library for shared usage. Try to reconfigure apache, making sure to use at least the following flags:

```
--enable-shared=max --enable-rule=SHARED_CORE
```

For more information, read the top-level Apache `INSTALL` file or the Apache DSO manual page.

5. When I run configure, it says that it can't find the include files or library for GD, gdbm, or some other package!

You can make the configure script looks for header files and libraries in non-standard locations by specifying additional flags to pass to the C preprocessor and linker, such as:

```
CPPFLAGS=-I/path/to/include LDFLAGS=-L/path/to/library ./configure
```

If you're using a csh-variant for your login shell (why?), it would be:

```
env CPPFLAGS=-I/path/to/include LDFLAGS=-L/path/to/library ./configure
```

6. When it is compiling the file `language-parser.tab.c`, it gives me errors that say *yytname undeclared*.

You need to update your version of Bison. You can find the latest version at http://www.gnu.org/software/bison/bison.html.

7. When I run make, it seems to run fine but then fails when it tries to link the final application complaining that it can't find some files.

Some old versions of make that don't correctly put the compiled versions of the files in the functions directory into that same directory. Try running **cp *.o functions** and then re-running **make** to see if that helps. If it does, you should really upgrade to a recent version of GNU make.

8. When linking PHP, it complains about a number of undefined references.

Take a look at the link line and make sure that all of the appropriate libraries are being included at the end. Common ones that you might have missed are '-ldl' and any libraries required for any database support you included.

If you're linking with Apache 1.2.x, did you remember to add the appropriate information to the EXTRA_LIBS line of the Configuration file and re-rerun Apache's Configure script? See the INSTALL file that comes with the distribution for more information.

Some people have also reported that they had to add '-ldl' immediately following `libphp4.a` when linking with Apache.

9. I can't figure out how to build PHP with Apache 1.3.

This is actually quite easy. Follow these steps carefully:

- Grab the latest Apache 1.3 distribution from http://www.apache.org/dist/httpd/.
- Ungzip and untar it somewhere, for example `/usr/local/src/apache-1.3`.
- Compile PHP by first running **./configure --with-apache=/<path>/apache-1.3** (substitute <path> for the actual path to your apache-1.3 directory.
- Type **make** followed by **make install** to build PHP and copy the necessary files to the Apache distribution tree.
- Change directories into to your `/<path>/apache-1.3/src` directory and edit the `Configuration` file. Add to the file: *AddModule modules/php4/libphp4.a*.
- Type: **./configure** followed by *make*.
- You should now have a PHP-enabled httpd binary!

Note: You can also use the new Apache *./configure* script. See the instructions in the *README.configure* file which is part of your Apache distribution. Also have a look at the `INSTALL` file in the PHP distribution.

10. I have followed all the steps to install the Apache module version on UNIX, and my PHP

scripts show up in my browser or I am being asked to save the file.

This means that the PHP module is not getting invoked for some reason. Three things to check before asking for further help:

- Make sure that the httpd binary you are running is the actual new httpd binary you just built. To do this, try running: */path/to/binary/httpd -l*

  If you don't see `mod_php4.c` listed then you are not running the right binary. Find and install the correct binary.

- Make sure you have added the correct Mime Type to one of your *Apache .conf* files. It should be: *AddType application/x-httpd-php3 .php3* (for PHP 3)

  or *AddType application/x-httpd-php .php* (for PHP 4)

  Also make sure that this AddType line is not hidden away inside a <Virtualhost> or <Directory> block which would prevent it from applying to the location of your test script.

- Finally, the default location of the Apache configuration files changed between Apache 1.2 and Apache 1.3. You should check to make sure that the configuration file you are adding the AddType line to is actually being read. You can put an obvious syntax error into your httpd.conf file or some other obvious change that will tell you if the file is being read correctly.

11. It says to use: *--activate-module=src/modules/php4/libphp4.a*, but that file doesn't exist, so I changed it to *--activate-module=src/modules/php4/libmodphp4.a* and it doesn't work!? What's going on?

Note that the `libphp4.a` file is not supposed to exist. The apache process will create it!

12. When I try to build Apache with PHP as a static module using *--activate-module=src/modules/php4/libphp4.a* it tells me that my compiler is not ANSI compliant.

This is a misleading error message from Apache that has been fixed in more recent versions.

13. When I try to build PHP using *--with-apxs* I get strange error messages.

There are three things to check here. First, for some reason when Apache builds the apxs Perl script, it sometimes ends up getting built without the proper compiler and flags variables. Find your apxs script (try the command **which apxs**), it's sometimes found in `/usr/local/apache/bin/apxs` or `/usr/sbin/apxs`. Open it and check for lines similar to these:

```
my $CFG_CFLAGS_SHLIB  = ' ';          # substituted via Makefile.tmpl
my $CFG_LD_SHLIB      = ' ';          # substituted via Makefile.tmpl
my $CFG_LDFLAGS_SHLIB = ' ';          # substituted via Makefile.tmpl
```

If this is what you see, you have found your problem. They may contain just spaces or other incorrect values, such as 'q()'. Change these lines to say:

```
my $CFG_CFLAGS_SHLIB  = '-fpic -DSHARED_MODULE'; # substituted via Makefile.tmpl
my $CFG_LD_SHLIB      = 'gcc';                   # substituted via Makefile.tmpl
my $CFG_LDFLAGS_SHLIB = q(-shared);              # substituted via Makefile.tmpl
```

The second possible problem should only be an issue on Red Hat 6.1 and 6.2. The apxs script Red Hat ships is broken. Look for this line:

```
my $CFG_LIBEXECDIR    = 'modules';         # substituted via APACI install
```

If you see the above line, change it to this:

```
my $CFG_LIBEXECDIR    = '/usr/lib/apache'; # substituted via APACI install
```

Last, if you reconfigure/reinstall Apache, add a **make clean** to the process after **./configure** and before **make**.

14. During make, I get errors in microtime, and a lot of *RUSAGE_* stuff.

During the **make** portion of installation, if you encounter problems that look similar to this:

```
microtime.c: In function `php_if_getrusage':
microtime.c:94: storage size of `usg' isn't known
microtime.c:97: `RUSAGE_SELF' undeclared (first use in this function)
microtime.c:97: (Each undeclared identifier is reported only once
microtime.c:97: for each function it appears in.)
microtime.c:103: `RUSAGE_CHILDREN' undeclared (first use in this function)
make[3]: *** [microtime.lo] Error 1
make[3]: Leaving directory `/home/master/php-4.0.1/ext/standard'
make[2]: *** [all-recursive] Error 1
make[2]: Leaving directory `/home/master/php-4.0.1/ext/standard'
make[1]: *** [all-recursive] Error 1
make[1]: Leaving directory `/home/master/php-4.0.1/ext'
make: *** [all-recursive] Error 1
```

Your system is broken. You need to fix your `/usr/include` files by installing a glibc-devel package that matches your glibc. This has absolutely nothing to do with PHP. To prove this to yourself, try this simple test:

```
$ cat >test.c <<X
#include <sys/resource.h>
X
$ gcc -E test.c >/dev/null
```

If that spews out errors, you know your include files are messed up.

15. When compiling PHP with MySQL, configure runs fine but during *make* I get an error similar to the following: *ext/mysql/libmysql/my_tempnam.o(.text+0x46): In function my_tempnam': /php4/ext/mysql/libmysql/my_tempnam.c:103: the use of tempnam' is dangerous, better use mkstemp'*, what's wrong?

First, it's important to realize that this is a *Warning* and not a fatal error. Because this is often the last output seen during *make*, it may seem like a fatal error but it's not. Of course, if you set your compiler to die on Warnings, it will. Also keep in mind that MySQL support is enabled by default.

16. I want to upgrade my PHP. Where can I find the ./configure line that was used to build my current PHP installation?

Either you look at config.nice file, in the source tree of your current PHP installation or, if this is not available, you simply run a

```
<?php phpinfo(); ?>
```

script. On top of the output the **./configure** line, that was used to build this PHP installation is shown.

17. When building PHP with the GD library it either gives strange compile errors or segfaults on execution.

Make sure your GD library and PHP are linked against the same depending libraries (e.g. libpng).

Capítulo 70. Uso de PHP

Esta sección reúne varios errores comunes que usted puede enfrentar cuando escribe scripts PHP.


```
<?php
function mi_funcion($argumento)
{
    echo $argumento + 10;
}
$variable = 10;
echo "mi_funcion($variable) = " . mi_funcion($variable);
?>
```



```
<pre>
<?php echo "Esta debe ser la primera l&iacute;nea."; ?>
<?php echo "Esto debe aparecer despu&eacute;s de la nueva l&iacute;nea anterior."; ?>
</pre>
```


1. Quisiera escribir un script PHP genérico que pudiera manejar datos que lleguen desde cualquier formulario. ¿Cómo averiguo qué variables del método POST se encuentran disponibles?

PHP ofrece numerosas variables predefinidas, como la super-global *$_POST*. Usted podría recorrer *$_POST* con un ciclo ya que se trata de una matriz asociativa con todos los valores POST. Por ejemplo, puede recorrer la matriz simplemente con foreach, realizar un chequeo por valores **empty ()**, e imprimirlos.

```
<?php
$vacio = $post = array();
foreach ($_POST as $nombre_var => $valor_var) {
    if (empty($valor_var)) {
        $vacio[$nombre_var] = $valor_var;
    } else {
        $post[$nombre_var] = $valor_var;
    }
}

print "<pre>";
if (empty($vacio)) {
    print "Ninguno de los valores POST est&aacute;n vac&iacute;os, se envi&oacute;:\n";
    var_dump($post);
} else {
    print "Tenemos " . count($vacio) . " valores vac&iacute;os\n";
    print "Se envi&oacute;:\n"; var_dump($post);
    print "Vac&iacute;os:\n";  var_dump($vacio);
    exit;
}
?>
```

> **Superglobals: Nota de disponibilidad:** Desde 4.1.0, están disponibles algunas matrices superglobales tales como *$_GET*, *$_POST*, y *$_SERVER*, etc. Para más información puede consultar la sección superglobals

2. Necesito convertir todas las comillas sencillas (') en una barra invertida seguida de una comilla sencilla (\'). ¿Cómo puedo hacer esto con una expresión regular? También me gustaría convertir " en \" y \ en \\.

La función **addslashes()** hace esto. Vea también **mysql_escape_string()**. También puede remover las barras invertidas con **stripslashes()**.

> **Nota de directiva: magic_quotes_gpc:** La directiva PHP magic_quotes_gpc tiene por defecto el valor *on*. Básicamente ejecuta **addslashes()** en los datos obtenidos por GET, POST, y COOKIE. Se puede usar **stripslashes()** para quitarlos.

3. Todos mis caracteres " se convierten en \" y mis ' se convierten en \', ¿cómo me deshago de todas esas barras invertidas indeseadas? ¿Cómo y porqué llegaron allí?

La función de PHP **stripslashes()** eliminará todas esas barras invertidas de su variable **string**. Lo más posible es que esas barras invertidas existen mágicamente por que la directiva de PHP magic_quotes_gpc está activa.

> **Nota de directiva: magic_quotes_gpc:** La directiva PHP magic_quotes_gpc tiene por defecto el valor *on*. Básicamente ejecuta **addslashes()** en los datos obtenidos por GET, POST, y COOKIE. Se puede usar **stripslashes()** para quitarlos.

4. Cuando hago lo siguiente, la salida se imprime en el orden ñocado:

```
<?php
function mi_funcion($argumento)
{
    echo $argumento + 10;
}
$variable = 10;
echo "mi_funcion($variable) = " . mi_funcion($variable);
?>
```

¿qué sucede?

Para poder usar los resultados de su función en una expresión (como ocurre al concatenar con otras cadenas, como en el ejemplo anterior), necesita devolver el valor mediante **return()**, no imprimirlo

con **echo()**.

5. ¿Hey, qué ha sucedido con mis saltos de línea?

```
<pre>
<?php echo "Esta debe ser la primera l&iacute;nea."; ?>
<?php echo "Esto debe aparecer despu&eacute;s de la nueva l&iacute;nea anterior."; ?>
</pre>
```

En PHP, el final de un bloque de código es, o bien "?>" o "?>\n" (en donde \n quiere decir una nueva línea). De modo que en el ejemplo anterior, las sentencias impresas con echo estarán en una línea, puesto que PHP omite los saltos de línea después del final del bloque. Esto quiere decir que usted necesita insertar un salto de línea extra después de cada bloque de código PHP para hacer que imprima una nueva línea.

¿Porqué hace esto PHP? Por que al dar formato a un documento HTML normal, este comportamiento por lo general hace su vida más simple, ya que no desea ese salto de línea, pero tendría que crear líneas extremadamente largas o de otra forma hacer ilegible el código fuente base para lograr ese efecto.

6. Recibo el mensaje 'Warning: Cannot send session cookie - headers already sent...' o 'Cannot add header information - headers already sent...'.

Las funciones **header()**, **setcookie()**, y las funciones de sesión necesitan agregar cabeceras a la secuencia de salida, pero las cabeceras sólo pueden ser enviadas antes del resto del contenido. No puede haber salida antes de usar éstas funciones, salida como HTML. La función **headers_sent()** revisa si su script ya ha enviado las cabeceras, y asimismo consulte las funciones de Control de Salida.

7. Necesito acceder a información directamente de las cabeceras de la petición. ¿Cómo puedo hacer esto?

La función **getallheaders()** hará esto si está ejecutando PHP como módulo de Apache. Así que, el siguiente segmento de código le mostrará todas las cabeceras de petición:

```
<?php
$cabeceras = getallheaders();
foreach ($cabeceras as $nombre => $contenido) {
    echo "cabeceras[$nombre] = $contenido<br />\n";
}
?>
```

Vea también **apache_lookup_uri()**, **apache_response_headers()**, y **fsockopen()**

8. Cuando intento usar autenticación con IIS, recibo el mensaje 'No Input file specified'.

El modelo de seguridad de IIS es la causa del problema aquí. Este es un inconveniente común a todos los programas CGI que corren bajo IIS. Una forma de evitar el problema es crear un archivo HTML plano (no interpretado por PHP) como página de entrada en un directorio autenticado. Luego usar una etiqueta META para redirigir a la página PHP, o tener un enlace hacia la página PHP. PHP reconocerá entonces la autenticación correctamente. Con el módulo ISAPI, esto no es un problema. Este inconveniente no debe afectar otros servidores web NT. Para más información, vea: http://support.microsoft.com/support/kb/articles/q160/4/22.asp y la sección del manual sobre Autenticación HTTP.

9. Mi script PHP funciona en IE y Lynx, pero en Netscape parte de mi salida está faltando. Cuando acciono "Ver código fuente" veo el contenido en IE pero no en Netscape.

Netscape es más estricto que IE en cuanto a etiquetas HTML (como tablas) se refiere. Pasar su salida HTML a través de un validador de HTML, como validator.w3.org, puede ser de ayuda. Por

ejemplo, una etiqueta </table> faltante puede ser la causa de éste problema.

Asimismo, tanto IE como Lynx ignoran los caracteres NUL (*\0*) en la secuencia HTML, mientras que Netscape no. La mejor manera de chequear esto es compilar la versión de línea de comandos de PHP (también conocida como la versión CGI) y ejecutar su script desde la línea de comandos. En *nix, envíe la salida a través de un pipe a *od -c* y busque caracteres *\0*. Si se encuentra en Windows necesita un editor o algún otro programa que le permita ver archivos binarios. Cuando Netscape ve un NUL en un archivo, típicamente no imprimirá nada más en aquella línea, mientras que tanto IE como Lynx si lo hacen.

10. ¿Cómo se supone que mezcle XML y PHP? ¡Se queja sobre mis etiquetas <?xml!

Para poder embeber etiquetas <?xml directamente en su código PHP, tendrá que deshabilitar las etiquetas cortas, definiendo la directiva PHP short_open_tags como *0*. No puede definir esta directiva con **ini_set()**. Independientemente del valor de short_open_tags, usted puede hacer algo como: *<?php echo '<?xml'; ?>*. El valor predeterminado para esta directiva es on.

11. ¿Cómo puedo usar PHP con FrontPage u otro editor HTML que insiste en mover mi código por todas partes?

Una de las cosas más fáciles es habilitar el uso de etiquetas ASP en su código PHP. Esto le permite usar el estilo de delimitadores de código ASP <% y %>. Algunos de los editores HTML populares gestionan éstas etiquetas de forma más inteligente (por ahora). Para habilitar las etiquetas estilo-ASP, necesita definir la variable `php.ini` asp_tags, o usar la directiva de Apache apropiada.

12. ¿En dónde puedo encontrar una lista completa de variables disponibles en PHP?

Consulte la página del manual sobre variables predefinidas, ya que allí se incluye una lista parcial de variables predefinidas disponibles para su script. Una lista completa de variables disponibles (y mucha más información) puede ser consultada al llamar la función **phpinfo()**. Asegúrese de leer la sección del manual sobre variables externas a PHP ya que allí se describen escenarios comunes para variables externas, como las provenientes de formularios HTML, Cookies y URLs.

> **register_globals: Nota importante:** Desde PHP 4.2.0 el valor por defecto de la directiva register_globals es *off*. La comunidad PHP anima a todos a no confiar en esta directiva y usar en su lugar superglobals.

13. ¿Cómo puedo generar archivos PDF sin usar las bibliotecas no-libres y comerciales ClibPDF y PDFLib? Quisiera algo que fuera gratuito y no requiera de bibliotecas PDF externas.

Existen algunas pocas alternativas escritas en PHP, como http://www.ros.co.nz/pdf/, http://www.fpdf.org/, http://www.gnuvox.com/pdf4php/, y http://www.potentialtech.com/ppl.php. También existe el módulo Panda.

14. Estoy tratando de acceder a una de las variables CGI estándar (como $DOCUMENT_ROOT o $HTTP_REFERER) en una función definida por el usuario, y parece que no la encuentra. ¿Qué está fallando?

Es importante darse cuenta de que la directiva PHP register_globals también afecta a variables de servidor y de entorno. Cuando register_globals = off (el valor predeterminado es off a partir de PHP 4.2.0), *$DOCUMENT_ROOT* no existirá. En su lugar, use *$_SERVER['DOCUMENT_ROOT']*. Si register_globals = on entonces las variables *$DOCUMENT_ROOT* y *$GLOBALS ['DOCUMENT_ROOT']* existirán también.

Si está seguro de que register_globals = on y se pregunta porqué *$DOCUMENT_ROOT* no se encuentra disponible al interior de funciones, es porque éste tipo de variables es como cualquier otra, y requeriría de una sentencia *global $DOCUMENT_ROOT* al interior de la función. Vea también la página del manual sobre los contextos de variables. Es recomendable escribir código bajo el modelo register_globals = off.

> **Superglobals: Nota de disponibilidad:** Desde 4.1.0, están disponibles algunas matrices superglobales tales como *$_GET*, *$_POST*, y *$_SERVER*, etc. Para más información puede consultar la sección superglobals

15. Algunas directivas de PHP pueden recibir también atajos de valores de bytes, en lugar de recibir sólamente valores de bytes tipo integer. ¿Cuáles son todas las opciones de atajos de valores de byte disponibles? ¿Y puedo usarlos por fuera de `php.ini`?

Las opciones disponibles son K (para Kilobytes) y M (para Megabytes), las cuales son insensibles a mayúsculas y minúsculas. Cualquier otro valor asume bytes. *1M* es igual a un Megabyte o *1048576* bytes. *1K* es igual a un Kilobyte o *1024* bytes. No debería usar éstas notaciones cortas por fuera de `php.ini`, en su lugar use un valor **integer** de bytes. Vea la documentación de **ini_get()** para un ejemplo sobre cómo convertir tales valores.

Capítulo 71. PHP and HTML

PHP and HTML interact a lot: PHP can generate HTML, and HTML can pass information to PHP. Before reading these faqs, it's important you learn how to retrieve variables from outside of PHP. The manual page on this topic includes many examples as well. Pay close attention to what *register_globals* means to you too.

1. What encoding/decoding do I need when I pass a value through a form/URL?

There are several stages for which encoding is important. Assuming that you have a **string** *$data*, which contains the string you want to pass on in a non-encoded way, these are the relevant stages:

- HTML interpretation. In order to specify a random string, you *must* include it in double quotes, and **htmlspecialchars()** the whole value.
- URL: A URL consists of several parts. If you want your data to be interpreted as one item, you *must* encode it with **urlencode()**.

Ejemplo 71-1. A hidden HTML form element

```
<?php
    echo "<input type=hidden value=\"" . htmlspecialchars($data) . "\">\n";
?>
```

> **Nota:** It is wrong to **urlencode()** *$data*, because it's the browsers responsibility to **urlencode()** the data. All popular browsers do that correctly. Note that this will happen regardless of the method (i.e., GET or POST). You'll only notice this in case of GET

request though, because POST requests are usually hidden.

Ejemplo 71-2. Data to be edited by the user

```
<?php
    echo "<textarea name=mydata>\n";
    echo htmlspecialchars($data)."\n";
    echo "</textarea>";
?>
```

Nota: The data is shown in the browser as intended, because the browser will interpret the HTML escaped symbols.

Upon submitting, either via GET or POST, the data will be urlencoded by the browser for transferring, and directly urldecoded by PHP. So in the end, you don't need to do any urlencoding/urldecoding yourself, everything is handled automagically.

Ejemplo 71-3. In an URL

```
<?php
    echo "<a href=\"" . htmlspecialchars("/nextpage.php?stage=23&data=" .
        urlencode($data)) . "\">\n";
?>
```

Nota: In fact you are faking a HTML GET request, therefore it's necessary to manually **urlencode()** the data.

Nota: You need to **htmlspecialchars()** the whole URL, because the URL occurs as value of an HTML-attribute. In this case, the browser will first un-**htmlspecialchars()** the value, and then pass the URL on. PHP will understand the URL correctly, because you **urlencoded()** the data.

You'll notice that the *&* in the URL is replaced by *&*. Although most browsers will recover if you forget this, this isn't always possible. So even if your URL is not dynamic, you *need* to **htmlspecialchars()** the URL.

2. I'm trying to use an <input type="image"> tag, but the *$foo.x* and *$foo.y* variables aren't available. *$_GET['foo.x']* isn't existing either. Where are they?

When submitting a form, it is possible to use an image instead of the standard submit button with a tag like:

```
<input type="image" src="image.gif" name="foo">
```

When the user clicks somewhere on the image, the accompanying form will be transmitted to the server with two additional variables: *foo.x* and *foo.y*.

Because *foo.x* and *foo.y* would make invalid variable names in PHP, they are automagically converted to *foo_x* and *foo_y*. That is, the periods are replaced with underscores. So, you'd access these variables like any other described within the section on retrieving variables from outside of PHP. For example, *$_GET['foo_x']*.

3. How do I create arrays in a HTML <form>?

To get your <form> result sent as an array to your PHP script you name the <input>, <select> or <textarea> elements like this:

```
<input name="MyArray[]">
<input name="MyArray[]">
<input name="MyArray[]">
<input name="MyArray[]">
```

Notice the square brackets after the variable name, that's what makes it an array. You can group the

elements into different arrays by assigning the same name to different elements:

```
<input name="MyArray[]">
<input name="MyArray[]">
<input name="MyOtherArray[]">
<input name="MyOtherArray[]">
```

This produces two arrays, MyArray and MyOtherArray, that gets sent to the PHP script. It's also possible to assign specific keys to your arrays:

```
<input name="AnotherArray[]">
<input name="AnotherArray[]">
<input name="AnotherArray[email]">
<input name="AnotherArray[phone]">
```

The AnotherArray array will now contain the keys 0, 1, email and phone.

> **Nota:** Specifying an arrays key is optional in HTML. If you do not specify the keys, the array gets filled in the order the elements appear in the form. Our first example will contain keys 0, 1, 2 and 3.

See also Array Functions and Variables from outside PHP.

4. How do I get all the results from a select multiple HTML tag?

The select multiple tag in an HTML construct allows users to select multiple items from a list. These items are then passed to the action handler for the form. The problem is that they are all passed with the same widget name. ie.

```
<select name="var" multiple>
```

Each selected option will arrive at the action handler as:

```
var=option1
var=option2
var=option3
```

Each option will overwrite the contents of the previous *$var* variable. The solution is to use PHP's "array from form element" feature. The following should be used:

```
<select name="var[]" multiple>
```

This tells PHP to treat *$var* as an array and each assignment of a value to var[] adds an item to the array. The first item becomes *$var[0]*, the next *$var[1]*, etc. The **count()** function can be used to determine how many options were selected, and the **sort()** function can be used to sort the option array if necessary.

Note that if you are using JavaScript the *[]* on the element name might cause you problems when you try to refer to the element by name. Use it's numerical form element ID instead, or enclose the variable name in single quotes and use that as the index to the elements array, for example:

```
variable = documents.forms[0].elements['var[]'];
```

5. How can I pass a variable from Javascript to PHP?

Since Javascript is (usually) a client-side technology, and PHP is (usually) a server-side technology, and since HTTP is a "stateless" protocol, the two languages cannot directly share variables.

It is, however, possible to pass variables between the two. One way of accomplishing this is to generate Javascript code with PHP, and have the browser refresh itself, passing specific variables back to the PHP script. The example below shows precisely how to do this -- it allows PHP code to capture screen height and width, something that is normally only possible on the client side.

```
<?php
if (isset($_GET['width']) AND isset($_GET['height'])) {
  // output the geometry variables
  echo "Screen width is: ". $_GET['width'] ."<br />\n";
  echo "Screen height is: ". $_GET['height'] ."<br />\n";
} else {
  // pass the geometry variables
  // (preserve the original query string
  //   -- post variables will need to handled differently)

  echo "<script language=\"javascript\">\n";
  echo "  location.href=\"${_SERVER['SCRIPT_NAME']}?${_SERVER['QUERY_STRING']}"
           . "&width=\" + screen.width + \"&height=\" + screen.height;\n";
  echo "</script>\n";
  exit();
}
?>
```

Capítulo 72. PHP and COM

PHP can be used to access COM and DCOM objects on Win32 platforms.

1. I have built a DLL to calculate something. Is there any way to run this DLL under PHP ?

If this is a simple DLL there is no way yet to run it from PHP. If the DLL contains a COM server you may be able to access it if it implements the IDispatch interface.

2. What does 'Unsupported variant type: xxxx (0xxxxx)' mean ?

There are dozens of VARIANT types and combinations of them. Most of them are already supported but a few still have to be implemented. Arrays are not completely supported. Only single dimensional indexed only arrays can be passed between PHP and COM. If you find other types that aren't supported, please report them as a bug (if not already reported) and provide as much information as available.

3. Is it possible manipulate visual objects in PHP ?

Generally it is, but as PHP is mostly used as a web scripting language it runs in the web servers context, thus visual objects will never appear on the servers desktop. If you use PHP for application scripting e.g. in conjunction with PHP-GTK there is no limitation in accessing and manipulating visual objects through COM.

4. Can I store a COM object in a session ?

No, you can't. COM instances are treated as resources and therefore they are only available in a single script's context.

5. How can I trap COM errors ?

Currently it's not possible to trap COM errors beside the ways provided by PHP itself (@, track_errors, ..), but we are thinking of a way to implement this.

6. Can I generate DLL files from PHP scripts like i can in Perl ?

No, unfortunately there is no such tool available for PHP.

7. What does 'Unable to obtain IDispatch interface for CLSID {xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}' mean ?

This error can have multiple reasons:

- the CLSID is wrong
- the requested DLL is missing
- the requested component doesn't implement the IDispatch interface

8. How can I run COM object from remote server ?

Exactly like you run local objects. You only have to pass the IP of the remote machine as second parameter to the COM constructor.

Make sure that you have set *com.allow_dcom=true* in your `php.ini`.

9. I get 'DCOM is disabled in C:\path...\scriptname.php on line 6', what can I do ?

Edit your `php.ini` and set *com.allow_dcom=true*.

10. Is it possible to load/manipulate an ActiveX object in a page with PHP ?

This has nothing to do with PHP. ActiveX objects are loaded on client side if they are requested by the HTML document. There is no relation to the PHP script and therefore there is no direct server side interaction possible.

11. Is it possible to get a running instance of a component ?

This is possible with the help of monikers. If you want to get multiple references to the same word instance you can create that instance like shown:

```
$word = new COM("C:\docs\word.doc");
```

This will create a new instance if there is no running instance available or it will return a handle to

the running instance, if available.

12. Is there a way to handle an event sent from COM object ?

Starting in PHP 4.3.0, you can define an event sink and bind it as shown in the example below. You can use **com_print_typeinfo()** to have PHP generate a skeleton for the event sink class.

Ejemplo 72-1. COM event sink example

```
<?php
class IEEventSinker {
  var $terminated = false;

  function ProgressChange($progress, $progressmax) {
    echo "Download progress: $progress / $progressmax\n";
  }

  function DocumentComplete(&$dom, $url) {
    echo "Document $url complete\n";
  }

  function OnQuit() {
    echo "Quit!\n";
    $this->terminated = true;
  }
}
$ie = new COM("InternetExplorer.Application");
$sink =& new IEEventSinker();
com_event_sink($ie, $sink, "DWebBrowserEvents2");
$ie->Visible = true;
$ie->Navigate("http://www.php.net");
while(!$sink->terminated) {
  com_message_pump(4000);
}
$ie = null;
?>
```

13. I'm having problems when trying to invoke a method of a COM object which exposes more than one interface. What can I do ?

The answer is as simple as unsatisfying. I don't know exactly but i think you can do nothing. If someone has specific information about this, please let me know :)

14. So PHP works with COM, how about COM+ ?

COM+ extends COM by a framework for managing components through MTS and MSMQ but there is nothing special that PHP has to support to use such components.

15. If PHP can manipulate COM objects, can we imagine to use MTS to manage components resources, in conjunction with PHP ?

PHP itself doesn't handle transactions yet. Thus if an error occurs no rollback is initiated. If you use components that support transactions you will have to implement the transaction management yourself.

Capítulo 73. PHP y otros lenguajes

PHP es el mejor lenguaje para programación web, ¿pero qué hay de los otros lenguajes?

1. ¿PHP vs. ASP?

1. ¿PHP vs. ASP?

ASP no es realmente un lenguaje como tal, es el acrónimo de Active Server Pages, el lenguaje usado en realidad para programar ASP es Visual Basic Script o JScript. El mayor inconveniente de ASP es que se trata de un sistema propietario que es usado nativamente sólo por Microsoft Internet Information Server (IIS). Esto limita su disponibilidad a servidores basados en Win32. Existe un par de proyectos en desarrollo que permiten que ASP corra en otros entornos y servidores web: InstantASP de Halcyon (comercial), Chili!Soft ASP de Chili!Soft (comercial). Se dice que ASP es un lenguaje más lento y pesado que PHP, y también menos estable. Algunas de las ventajas de ASP consisten en que debido a que usa principalmente VBScript, es relativamente simple tratar con el lenguaje si usted ya conoce cómo programar en Visual Basic. El soporte de ASP también se encuentra habilitado por defecto en el servidor IIS, facilitando su instalación y ejecución. Los componentes integrados en ASP son bastante limitados, de modo que si necesita usar características "avanzadas", como interactuar con servidores FTP, necesita comprar componentes adicionales.

2. ¿Existe software de conversión de ASP a PHP?

Si, la herramienta de lado del servidor asp2php es una de las más conocidas, así como ésta opción de lado del cliente.

3. ¿PHP vs. Cold Fusion?

Comúnmente se dice que PHP es más rápido y eficiente para tareas complejas de programación y cuando se desea probar ideas nuevas. Asimismo, PHP se considera por lo general más estable y menos intensivo en el uso de recursos. Cold Fusion posee un mejor gestor de errores, asi como abstracciones de bases de datos y procesamiente de fechas, aunque la abstracción de bases de datos es parte de PHP 4. Otra de las cosas que se lista como una de las fortalezas de Cold Fusion es su excelente motor de búsqueda, pero se ha dicho que un motor de búsqueda no es algo que deba ser incluido en un lenguaje de scripting orientado a web. PHP corre en casi cualquier plataforma que existe; Cold Fusion sólo se encuentra disponible en Win32, Solaris, Linux y HP/UX. Cold Fusion posee un buen entorno integrado de desarrollo, y es generalmente más sencillo para principiantes, mientras que PHP requiere inicialmente de un mayor conocimiento de programación. Cold Fusion está diseñado hacia personas sin experiencia en programación, mientras que PHP se concentra en los programadores.

Un excelente resumen sobre este tema, escrito por Michael J Sheldon, ha sido publicado en la lista de correo de PHP. Puede encontrarse una copia en http://marc.theaimsgroup.com/?l=php-general&m=95602167412542&w=1.

4. ¿PHP vs. Perl?

La mayor ventaja de PHP sobre Perl es que PHP fue diseñado para desarrollo de scripts orientados a web, mientras que Perl fue diseñado para hacer muchas más cosas y debido a esto, se hace muy complicado. La flexibilidad / complejidad de Perl facilitan la escritura de código que otro autor / programador puede encontrar muy difícil de entender. PHP posee un formato menos confuso y más estricto, sin perder flexibilidad. Con PHP es más fácil la integración con HTML que con Perl. PHP cuenta con prácticamente toda la funcionalidad 'buena' de Perl: construcciones del lenguaje, sintaxis y demás, sin hacerlo tan complicado como Perl puede llegar a ser. Perl es un lenguaje muy usado y auténtico, ha estado vigente desde finales de los ochentas, pero PHP está madurando muy rápidamente.

Capítulo 74. Migración de PHP 2 a PHP 3

PHP ya tiene una larga historia tras él: Las legendarias versiones PHP 1.0, PHP/FI, PHP 3.0 y PHP 4.0.

1. ¿Migración de PHP 2 a PHP 3?

Ya no se ofrece soporte para PHP/FI 2.0. Por favor refiérase a la sección apropiada del manual para más información sobre la migración desde PHP/FI 2.0.

Si aun está trabajando con PHP 2, le recomendamos *fuertemente* que se actualice directamente a PHP 4.

Capítulo 75. Migración de PHP 3 a PHP 4

PHP ya tiene una larga historia tras él: Las legendarias versiones PHP 1.0, PHP/FI, PHP 3.0 y PHP 4.0.

1. Migración de PHP 3 a PHP 4

PHP 4 fue diseñado para ser tan compatible con versiones anteriores como fuera posible, y muy poco de su funcionalidad fue descontinuado en el proceso. Si se encuentra realmente inseguro sobre la compatibilidad, debería instalar PHP 4 en un entorno de prueba y correr sus scripts allí.

Asimismo, vea el apéndice de migración apropiado de este manual.

2. ¿Funcionan las sesiones en PHP 3?

Aun cuando el soporte nativo de sesiones no existía en PHP 3, existen aplicaciones desarrolladas por terceros que ofrecían (y aun lo hacen) la funcionalidad de las sesiones. El método más común era mediante el uso de PHPLIB.

3. ¿Funciones incompatibles?

Dado que PHP 4 es básicamente una nueva versión escrita desde ceros del motor de PHP completo, hubo unas pocas funciones que fueron alteradas, y después de todo, solo algunas de las más exóticas.

Capítulo 76. Migrating from PHP 4 to PHP 5

This faq section will help you migrate from PHP 4 to PHP 5.

1. Migrating from PHP 4 to PHP 5

Although PHP 5 offers many new features, it's designed to be as compatible with earlier versions of PHP as possible with little functionality being broken in the process.

Be sure to read the appropriate PHP 5 migration appendix of this manual as it contains even more information on the topic of migrating to PHP 5.

2. Does MySQL work in PHP 5? It seemed to have disappeared.

MySQL is supported with the only change being that MySQL support is no longer enabled by *default* in PHP 5. This essentially means that PHP doesn't include the *--with-mysql* option in the configure line so that you must now manually do this when compiling PHP. Windows users will edit `php.ini` and enable the `php_mysql.dll` DLL as in PHP 4 no such DLL existed, it was simply built into your Windows PHP binaries.

Also, the MySQL client libraries are no longer bundled with PHP. More details on this topic are covered in the following FAQ and be sure to read the MySQL section for details on installing MySQL. An example configure line would be *--with-mysql=/usr* while Windows users will need the `libmySQL.dll` available to the system.

3. I hear PHP 5 has an entirely new OOP model, will my existing OOP code work? Where do I find information on these new OOP features?

The main change in PHP 5 is to the OOP model as PHP 5 now uses the *Zend Engine 2.0*. The zend.ze1_compatibility_mode directive enables compatability with the Zend Engine 1.0 (PHP 4).

The new OOP model is documented in the OOP language reference and OOP migration appendix sections.

4. So besides the new OOP model, what else has changed in PHP 5? Also, is there a PHP 5 specific version of the PHP manual?

Few other changes exist, see the migration 5 appendix for details. There won't be a PHP 5 specific version of the manual as the bulk of PHP remains the same.

Capítulo 77. Preguntas Varias

Puede que existan algunas preguntas que no podemos colocar en otras categorías. Este es el lugar en donde puede encontrarlas.

1. ¿Cómo puedo manipular los manuales comprimidos mediante bz2 en Windows?

Si no cuenta con una herramienta de archivación que pueda manejar archivos bz2, descargue la herramienta de línea de comandos de RedHat (por favor refiérase a la inforamción presentada más adelante).

Si no desea usar una herramienta de línea de comandos, puede probar herramientas gratuitas como Stuffit Expander, UltimateZip, 7-Zip, o Quick Zip. Si dispone de herramientas como WinRAR o Power Archiver, puede descomprimir fácilmente archivos bz2 con ellas. Si usa Total Commander (anteriormente Windows Commander), un módulo adicional para ese programa se encuentra disponible de forma gratuita desde el sitio de Total Commander.

La herramienta bzip2 de línea de comandos por Redhat:

Los usuarios de Win2k Sp2 deben obtener la versión más reciente, 1.0.2, todos los demás usuarios de Windows deben obtener la versión 1.00. Después de la descarga, renombre el ejecutable a bzip2.exe. Para mayor conveniencia, colóquelo en un directorio que sea parte de sus rutas predeterminadas, p.ej. C:\Windows, en donde C representa la unidad en donde se encuentra su instalación de windows.

Nota: lang representa su lenguaje, y x el formato deseado, p.ej: pdf. Para descomprimir el archivo php_manual_lang.x.bz2 siga las siguientes instrucciones:

- abra una ventana con el intérprete de comandos
- cambie de directorio hacia la carpeta en donde almacenó el archivo php_manual_lang.x.bz2 descargado
- invoque bzip2 -d php_manual_lang.x.bz2, extrayendo de este modo php_manual_lang.x en la misma carpeta

En caso de que haya descargado el archivo php_manual_lang.tar.bz2 con varios archivos html en su interior, el procedimiento es el mismo. La única diferencia es que recibe un archivo php_manual_lang.tar. Se conoce que el formato tar es tratado por la mayoría de archivadores en Windows, como por ejemplo WinZip.

2. ¿Qué significa un signo & al lado de un argumento en la declaración de una función, como p.ej. asort()?

Quiere decir que el argumento es pasado por referencia y es probable que la función modifique el argumento de acuerdo con la documentación. Sólo puede pasar variables de este modo y no necesita pasarlas con el signo & en la llamada de la función (tal cosa es considerada obsoleta).

X. Apéndices

Tabla de contenidos

Apéndice A. Historia de PHP y proyectos relacionados

PHP ha recorrido un largo camino en los últimos años. Crecer hasta ser uno de los más importantes lenguajes de programación en entornos Web no ha sido tarea fácil. Aquellos interesados en ver brevemente cómo creció PHP hasta lo que es hoy en día, sigan leyendo. Antiguas versiones de PHP están disponibles en el Museo PHP.

Historia de PHP

PHP/FI

PHP es el heredero de un producto anterior, llamado PHP/FI. PHP/FI fue creado por Rasmus Lerdorf en 1995, inicialmente como un simple conjunto de scripts de Perl para controlar los accesos a su trabajo online. Llamó a ese conjunto de scripts 'Personal Home Page Tools'. Según se requería más funcionalidad, Rasmus fue escribiendo una implementación C mucho mayor, que era capaz de comunicarse con bases de datos, y permitía a los usuarios desarrollar sencillas aplicaciones Web dinámicas. Rasmus eligió liberar el código fuente de PHP/FI para que cualquiera pudiese utilizarlo, así como arreglar errores y mejorar el código.

PHP/FI, que se mantuvo para páginas personales y como intérprete de formularios, incluía algunas de las funcionalidads básicas de PHP tal y como lo conocemos hoy. Tenía variables como las de Perl, interpretación automática de variables de formulario y sintaxis embebida HTML. La sintaxis por sí misma era similar a la de Perl, aunque mucho más limitada, simple y algo inconsistente.

Por 1997, PHP/FI 2.0, la segunda escritura de la implementación en C, tuvo un seguimiento estimado de varios miles de usuarios en todo el mundo, con aproximadamente 50.000 dominios informando que lo tenían instalado, sumando alrededor del 1% de los dominios de Internet. Mientras había mucha gente contribuyendo con bits de código a este proyecto, era todavía en su mayor parte el proyecto de una sola persona.

PHP/FI 2.0 no se liberó oficialmente hasta Noviembre de 1997, después de gastar la mayoría de su vida en desarrollos beta. Fue sucedido en breve tiempo por las primeras versiones alfa de PHP 3.0.

PHP 3

PHP 3.0 era la primera versión que se parecía fielmente al PHP tal y como lo conocemos hoy en día. Fue creado por Andi Gutmans y Zeev Zuraski en 1997 reescribiéndolo completamente, después de que encontraran que PHP/FI 2.0 tenía pocas posibilidades para desarrollar una aplicación comercial que estaban desarrollando para un projecto universitario. En un esfuerzo para cooperar y empezar a construir sobre la base de usuarios de PHP/FI existente, Andi, Rasmus y Zeev decidieron cooperar y anunciar PHP 3.0 como el sucesor oficial de PHP/FI 2.0, interrumpiéndose en su mayor parte el desarrollo de PHP/FI 2.0.

Una de las mejores características de PHP 3.0 era su gran extensibilidad. Además de proveer a los usuarios finales de una sólida infraestructura para muchísimas bases de datos, protocolos y APIs, las características de extensibilidad de PHP 3.0 atrajeron a docenas de desarrolladores a unirse y enviar nuevos módulos de extensión. Sin duda, ésta fue la clave del enorme éxito de PHP 3.0. Otras características clave introducidas en PHP 3.0 fueron el soporte de sintáxis orientado a objetos y una sintáxis de lenguaje mucho más potente y consistente.

Todo el nuevo lenguaje fue liberado bajo un nuevo nombre, que borraba la implicación de uso personal limitado que tenía el nombre PHP/FI 2.0. Se llamó 'PHP' a secas, con el significado de ser un acrónimo recursivo - PHP: Hypertext Preprocessor.

A finales de 1998, PHP creció hasta una base de instalación de decenas de millares de usuarios (estimados) y cientos de miles de sitios Web informando de su instalación. En su apogeo, PHP 3.0 estaba instalado en aproximadamente un 10% de los servidores Web en Internet.

PHP 3.0 se liberó oficialmente en Junio de 1998, después de haber gastado unos 9 meses en pruebas públicas.

PHP 4

En el invierno de 1998, poco después del lanzamiento oficial de PHP 3.0, Andi Gutmans y Zeev Suraski comenzaron a trabajar en la reescritura del núcleo de PHP. Los objetivos de diseño fueron mejorar la ejecución de aplicaciones complejas, y mejorar la modularidad del código base de PHP. Estas aplicaciones se hicieron posibles por las nuevas características de PHP 3.0 y el apoyo de una gran variedad de bases de datos y APIs de terceros, pero PHP 3.0 no fue diseñado para el mantenimiento tan complejo de aplicaciones eficientemente.

El nuevo motor, apodado 'Motor Zend' (comprimido de sus apellidos, Zeev y Andi), alcanzó estos objetivos de diseño satisfactoriamente, y se introdujo por primera vez a mediados de 1999. PHP 4.0, basado en este motor, y acoplado con un gran rango de nuevas características adicionales, fue oficialmente liberado en Mayo de 2000, casi dos años después que su predecesor, PHP 3.0. Además de la mejora de ejecución de esta versión, PHP 4.0 incluía otras características clave como el soporte para la mayoría de los servidores Web, sesiones HTTP, buffers de salida, formas más seguras de controlar las entradas de usuario y muchas nuevas construcciones de lenguaje.

PHP 4 es actualmente la última versión liberada de PHP. Ya se está trabajando en modificar y mejorar el motor Zend para integrar las características que se diseñarían para PHP 5.0.

Hoy, se estima que PHP es usado por cientos de miles de programadores y muchos millones de sitios informan que lo tienen instalado, sumando más del 20% de los dominios en Internet.

El equipo de desarrollo de PHP incluye docenas de programadores, así como otras docenas de personas trabajando en proyectos relacionados con PHP como PEAR y el proyecto de documentación.

PHP 5

El futuro de PHP está dirigido por su núcleo, el motor Zend. PHP 5 incluira el nuevo motor Zend 2.0. Para obtener más información sobre este motor, consultar su página web.

Historia de los proyectos relacionados con PHP

PEAR

PEAR, el PHP Extension and Application Repository (originalmente, PHP Extension and Add-on Repository) es la versión de clases de creación de PHP, y puede crecer en el futuro para ser una de las vías clave para distribuir tanto PHP como extensiones PHP basados en C entre desarrolladores.

PEAR nació de las discusiones mantenidas en el PHP Developers' Meeting (PDM) transcurrido en Enero de 2000 en Tel Aviv. Fue creado por Stig S. Bakken, y lo dedicó a su primogénita, Malin Bakken.

Hasta principios de 2000, PEAR fue creciendo hasta ser un gran y significativo proyecto con un gran número de programadores trabajando en la implementación común, funcionalidad reutilizable para el beneficio de toda la comunidad PHP. PEAR hoy incluye una gran variedad de clases de infraestructura para acceso a bases de datos, caché de contenido, cálculos matemáticos, comercio electrónico y mucho más.

Más información sobre PEAR se puede encontrar en el manual de PEAR.

Iniciativa para la Garantía de Calidad de PHP

La Iniciativa para la Garantía de Calidad de PHP se configuró en el verano de 2000 en respuesta a los que criticaban que las versiones de PHP se liberaban sin que fueran comprobadas suficientemente para entornos de producción. El equipo ahora consiste en un grupo central de desarrolladores con un buen entendimiento del código base de PHP. Estos desarrolladores gastan mucho tiempo localizando y solucionando problemas con PHP. Además, hay muchos miembros de otros equipos que testean y prueban estas soluciones usando una gran variedad de plataformas.

PHP-GTK

PHP-GTK es la solución PHP para escribir las aplicaciones GUI del lado del cliente. Andrei Zmievski recuerda la planificación y la creación del proceso de PHP-GTK:

> Programar GUI siempre ha estado entre mis intereses, y he encontrado que Gtk+ es una

herramienta muy buena, salvo que programar con ella en C es algo tedioso. Tras presenciar las implementaciones de PyGtk y GTK-Perl, decidí ver si PHP se podría hacer con la interfaz Gtk+, incluso mínimamente. En Agosto de 2000 empecé a tener más tiempo libre, con lo que comencé a experimentar. Mi principal guía fue la implementación PyGtk con características completas bastante buenas y con un buen interfaz orientado a objetos. James Henstridge, el autor de PyGtk, proveyó mucha ayuda adicional durante esos estados iniciales.

Escribir las interfaces de todas las funciones Gtk+ estaba fuera de cuestión, por lo que mantuve la idea de generador de código, similar a cómo PyGtk lo hizo. El generador de código es un programa PHP que lee un conjunto de ficheros .defs que contienen las clases Gtk+, constantes e información de métodos y genera código C que interactúa PHP con ellos. Lo que no se puede generar automáticamente puede escribirse a mano en ficheros .overrides.

Trabajar en el generador de código y la infraestructura llevó algo de tiempo, porque pude dedicar poco tiempo a PHP-GTK durante el otoño de 2000. Después mostré PHP-GTK a Frank Kromann, que se interesó y empezó a ayudarme con el trabajo del generador de código y con la implementación para Win32. Cuando escribimos el primer programa Hola Mundo y funcionó, fue extremadamente excitante. Llevó un par de meses más llevar el proyecto a una condición presentable y la versión inicial se liberó el 1 de Marzo de 2001. La historia rápidamente llegó a SlashDot.

Sintiendo que PHP-GTK podría extenderse, configuré listas de correo separadas y repositorios CVS para ello, así como el sitio web gtk.php.net con la ayuda de Colin Viebrock. La documentación también tuvo que hacerse y James Moore llegó para ayudar en esto.

Desde su lanzamiento PHP-GTK fue ganando popularidad. Tenemos nuestro propio equipo de documentación, el manual sigue mejorando, la gente ha comenzado a escribir extensiones para PHP-GTK, y con ello más y mejores aplicaciones.

Libros sobre PHP

Según crecía PHP, empezo a ser reconocido como una popular plataforma de desarrollo web. Una de las más interesantes formas de ver esta tendencia era observando los libros sobre PHP que han ido llegando a lo largo de los años.

Por lo que nosotros sabemos, el primer libro dedicado a PHP fue 'PHP - tvorba interaktivních internetovÃƒÂ½ch aplikací' - un libro checo publicado en 1999, cuyo autor fue Jirka Koseksiendo. Al mes siguiente le siguio un libro alemán cuyos autores fueron Egon Schmid, Christian Cartus y Richard Blume. El primer libro en inglés sobre PHP se publicó poco después, y fue 'Core PHP Programming' de Leon Atkinson. Los dos libros cubrían PHP 3.0.

Mientras estos libros fueron los primeros de su tipo - fueron seguidos por un gran número de libros de una multitud de autores y editores. ¡Existen más 40 libros en inglés, 50 libros en alemán, y más de 20 libros en francés! Además, se pueden encontrar libros sobre PHP en la mayoría de las demás lenguas, incluyendo español, coreano, japonés y hebreo.

Evidentemente, este gran número de libros, escritos por diferentes autores, publicados por muchos editores, y su disponibilidad en tantas lenguas - son un fuerte testimonio del éxito mundial de PHP.

Publicaciones sobre PHP

Por lo que sabemos, el primer artículo sobre PHP en una revista impresa se publicó en la version checa de Computerworld en la primavera de 1998, y cubría PHP 3.0. Como con los libros, fue el primero en una serie de muchos artículos publicados sobre PHP en varias revistas importantes.

Artículos sobre PHP han aparecido en Dr. Dobbs, Linux Enterprise, Linux Magazine y muchas otras. Artículos sobre migración de aplicaciones basadas en PHP bajo Windows ¡han aparecido incluso en el MSDN de Microsoft!

Apéndice B. Migración desde PHP 4 a PHP 5

Qué ha cambiado en PHP 5

PHP 5 y el Motor Zend 2 integrado han mejorado significativamente el rendimiento y las capacidades de PHP, al mismo tiempo que se han tomado precauciones para evitar al máximo incompatibilidades con el código existente. Por lo tanto, migrar su código desde PHP 4 a la versión 5 debe resultar bastante fácil. La mayoría de código PHP 4 existente debe estar listo para correr sin modificaciones, pero aun es buena idea que conozca sobre las pocas diferencias y tome sus precauciones para probar el código antes de actualizar versiones en entornos en producción.

Cambios Incompatibles con Versiones Anteriores

Aunque la mayoría de código PHP 4 existente debe correr sin modificaciones, es importante que preste atención a los siguientes cambios incompatibles con versiones anteriores:

- Existen algunas palabras clave nuevas.
- **strrpos()** y **strripos()** ahora usan la cadena entera como aguja.
- El uso de índices de cadena inválidos generan errores de tipo **E_ERROR** en lugar de **E_WARNING**.
- Se modificó **array_merge()** para que acepte únicamente matrices. Si una variable de un tipo diferente a matriz es pasada, se genera un error de tipo **E_WARNING** para cada uno de esos parámetros. Tenga cuidado, ya que su código puede comenzar a emitir anuncios **E_WARNING** de la nada.
- La variable de servidor PATH_TRANSLATED ya no es definida de forma implícita bajo la SAPI de Apache2, a diferencia del comportamiento de PHP 4, en donde se define con el mismo valor de la variable de servidor SCRIPT_FILENAME cuando no se define por Apache. Este cambio fue realizado para seguir la especificación CGI. Por favor refiérase al bug #23610 para más información.
- La constante **T_ML_COMMENT** ya no es definida por la extensión Tokenizer. Si el valor de error_reporting es **E_ALL**, PHP generará una noticia. Aunque la constante **T_ML_COMMENT** nunca fue usada, estaba definida en PHP 4. Tanto en PHP 4 como en PHP 5 // y /* */ son resueltos como la constante **T_COMMENT**. Sin embargo, los comentarios estilo PHPDoc /**

*/, que son interpretados a partir de PHP 5, se reconocen como **T_DOC_COMMENT**.

- $_SERVER debe poblarse con argc y argv si variables_order incluye "S". Si ha configurado su sistema específicamente para no crear $_SERVER, entonces por supuesto que no aparecerá. El cambio fue hecho para lograr que argc y argv siempre estuvieran disponibles en la versión CLI independientemente del parámetro variables_order. Es decir, ahora la versión CLI define siempre las variables globales $argc y $argv.

- Un objeto sin propiedades ya no es considerado "vacío".

- En algunos casos, las clases deben ser declaradas antes de ser usadas. Esto sólo ocurre si algunas de las nuevas características de PHP 5 son usadas. De otro modo el comportamiento antiguo se conserva.

- **get_class()**, , **get_parent_class()** y **get_class_methods()** devuelven ahora el nombre de las clases/métodos como ellos fueron declarados (siguiendo las mayúsculas y minúsculas) lo cual puede llevar a problemas en scripts viejos que dependían en el comportamiento anterior (el nombre de la clase/método era devuelto en minúsculas siempre). Una posible solución es buscar las funciones mencionadas en todos sus scripts y usar **strtolower()**.

  Este cambio en la sensibilidad a minúsculas y mayúsculas se aplica también a las constantes predefinidas mágicamente **__CLASS__**, **__METHOD__**, y **__FUNCTION__**. Los valores son devueltos exactamente como son declarados (sensibles a mayúsculas y minúsculas).

- **ip2long()** ahora devuelve **FALSE** cuando una dirección IP inválida es pasada como argumento a la función, y no *-1*.

- Si hay funciones definidas en el archivo incluido, éstas pueden ser usadas en el archivo principal, sin importar que estén antes de una sentencia **return()** o después. Si el archivo es incluido dos veces, PHP 5 produce un error fatal ya que las funciones ya han sido declaradas, mientras que PHP 4 no se queja al respecto. Se recomienda usar **include_once()** en lugar de revisar si el archivo ya había sido incluido y retornar condicionalmente al interior del archivo incluido.

- **include_once()** y **require_once()** primero normalizan la ruta del archivo incluido en Windows de modo que incluir A.php y a.php incluyen el archivo solo una vez.

Ejemplo B-1. strrpos() y strripos() ahora usan la cadena entera como aguja

```
<?php
var_dump(strrpos('ABCDEF','DEF')); //int(3)

var_dump(strrpos('ABCDEF','DAF')); //bool(false)
?>
```

Ejemplo B-2. Un objeto sin propiedades ya no es considerado "vacío"

```
<?php
class prueba { }
$p = new prueba();

var_dump(empty($p)); // echo bool(false)

if ($p) {
    // Se ejecuta
}
?>
```

Ejemplo B-3. En algunos casos, las clases deben declararse antes de ser usadas

```
<?php

// funciona sin errores:
$a = new a();
class a {
}


// gerena un error:
$a = new b();

interface c{
}
class b implements c {
}

?>
```

CLI y CGI

En PHP 5, se efectuaron algunos cambios en el entorno CLI y los nombres de archivo CGI. En PHP 5, la versión CGI fue renombrada a `php-cgi.exe` (previamente `php.exe`) y la versión CLI ahora existe en el directorio principal (previamente era `cli/php.exe`).

En PHP 5 se introdujo también un nuevo modo: `php-win.exe`. Este es igual a la versión CLI, excepto que php-win no genera ninguna salida y por lo tanto no provee una consola (no aparece una "caja de dos" en la pantalla"). Este comportamiento es similar a php-gtk.

En PHP 5, la versión CLI siempre define las variables globales *$argv* y *$argc*, independientemente a cualquier directiva en `php.ini`. Incluso teniendo el parámetro register_argc_argv como *off* no tendrá efecto en la versión CLI.

Vea también la referencia de la línea de comandos.

Migración de Archivos de Configuración

Dado que los módulos ISAPI cambiaron sus nombres, de php4xxx a php5xxx, es necesario hacer algunos cambios en los archivos de configuración. También ocurrieron cambios en los nombres de archivos CLI y CGI. Por favor refiérase a la sección correspondiente para más información.

Migrar la configuración de Apache es extremadamente simple. Consulte el ejemplo a continuación para verificar el cambio que necesita realizar:

Ejemplo B-4. Migración de archivos de configuración de Apache para PHP 5

```
# cambie esta linea:
LoadModule php4_module /php/sapi/php4apache2.dll

# por esta:
LoadModule php5_module /php/php5apache2.dll
```

Si su servidor web está usando PHP en modo CGI, debe notar que la versión CGI ha cambiado su nombre de `php.exe` a `php-cgi.exe`. En apache, debe hacer algo como lo siguiente:

Ejemplo B-5. Migración de archivos de configuración de Apache para PHP 5, modo CGI

```
# cambie esta linea:
Action application/x-httpd-php "/php/php.exe"

# por esta:
Action application/x-httpd-php "/php/php-cgi.exe"
```

En otros servidores web necesita cambiar ya sea el nombre de archivo CGI o del módulo ISAPI.

Nuevas Funciones

En PHP 5 aparecen algunas funciones nuevas. A continuación se presenta una lista de ellas:

Matrices:

- **array_combine()** - Crea una matriz usando una matriz para las claves y otra para sus valores
- **array_diff_uassoc()** - Computa la diferencia de matrices con un chequeo adicional de índices, el cual es realizado con una llamada de retorno entregada por el usuario
- **array_udiff()** - Computa la diferencia de matrices usando una llamada de retorno para la comparación de datos
- **array_udiff_assoc()** - Computa la diferencia de matrices con un chequeo adicional de índices. Los datos son comparados usando una llamada de retorno
- **array_udiff_uassoc()** - Computa la diferencia de matrices con un chequeo adicional de índices. Los datos son comparados usando una llamada de retorno. El chequeo de índices es realizado usando una llamada de retorno
- **array_walk_recursive()** - Aplica una función de usuario recursivamente a cada miembro de una matriz
- **array_uintersect_assoc()** - Computa la intersección de matrices con un chequeo adicional de índices. Los datos son comparados usando una llamada de retorno
- **array_uintersect_uassoc()** - Computa la intersección de matrices con un chequeo adicional de índices. Tanto los datos como los índices son comparados usando llamadas de retorno
- **array_uintersect()** - Computa la intersección de matrices. Los datos son comparados usando una llamada de retorno

InterBase:

- **ibase_affected_rows()** - Devuelve el número de filas que fueron afectadas por la consulta anterior
- **ibase_backup()** - Inicia una labor de respaldo en el administrador de servicios y retorna inmediatamente
- **ibase_commit_ret()** - Aplica una transacción sin cerrarla
- **ibase_db_info()** - Solicita estadísticas sobre una base de datos

- **ibase_drop_db()** - Elimina una base de datos
- **ibase_errcode()** - Devuelve un código de error
- **ibase_free_event_handler()** - Cancela un gestor de eventos registrado
- **ibase_gen_id()** - Incrementa el generador nombrado y devuelve su nuevo valor
- **ibase_maintain_db()** - Ejecuta un comando de mantenimiento en el servidor de bases de datos
- **ibase_name_result()** - Asigna un nombre a un conjunto de resultados
- **ibase_num_params()** - Devuelve el número de parámetros en una consulta preparada
- **ibase_param_info()** - Devuelve información sobre un parámetro en una consulta preparada
- **ibase_restore()** - Inicia una labor de rescate en el administrador de servicios y retorna inmediatamente
- **ibase_rollback_ret()** - Revierte una transacción y conserva su contexto
- **ibase_server_info()** - Solicita estadísticas sobre una base de datos
- **ibase_service_attach()** - Conectarse al administrador de servicios
- **ibase_service_detach()** - Desconectarse del administrador de servicios
- **ibase_set_event_handler()** - Registrar una llamada de retorno a ser usada cuando se publican eventos
- **ibase_wait_event()** - Esperar a que un evento sea publicado por la base de datos

iconv:

- **iconv_mime_decode()** - Decodifica un campo de cabecera MIME
- **iconv_mime_decode_headers()** - Decodifica múltiples campos de cabecera MIME de una vez
- **iconv_mime_encode()** - Compone un campo de cabecera MIME
- **iconv_strlen()** - Devuelve el conteo de caracteres de la cadena
- **iconv_strpos()** - Encuentra la posición de la primera ocurrencia de una aguja al interior de un pajar
- **iconv_strrpos()** - Encuentra la última ocurrencia de una aguja al interior del rango especificado del pajar
- **iconv_substr()** - Recorta parte de una cadena

Secuencias:

- **stream_copy_to_stream()** - Copia datos de una secuencia a otra
- **stream_get_line()** - Obtiene una línea de un recurso tipo secuencia hasta cierto límite indicado
- **stream_socket_accept()** - Acepta una conexión sobre un socket creado por **stream_socket_server()**
- **stream_socket_client()** - Abre una conexión de socket de Internet Abierto o Dominio Unix
- **stream_socket_get_name()** - Recupera el nombre de sockets locales o remotos
- **stream_socket_recvfrom()** - Recibe datos de un socket, conectado o no
- **stream_socket_sendto()** - Envia un mensaje a un socket, bien esté conectado o no
- **stream_socket_server()** - Crea un socket de servidor de Internet o de dominio Unix

Fecha y hora:

- **idate()** - Dar formato de entero a una fecha/hora local
- **date_sunset()** - Hora de crepúsculo para un día y ubicación determinados
- **date_sunrise()** - Hora del alba para un día y ubicación determinados
- **time_nanosleep()** - Espera por un número de segundos y nano-segundos

Cadenas:

- **str_split()** - Convierte una cadena en una matriz
- **strpbrk()** - Busca una cadena por un conjunto cualquiera de caracteres
- **substr_compare()** - Comparación segura con material binario, opcionalmente insensitiva a mayúsculas y minúsculas, de dos cadenas a partir de un desplazamiento, y hasta un límite da caracteres

Other:

- **convert_uudecode()** - decodificar una cadena con el algoritmo uuencode
- **convert_uuencode()** - Codificar una cadena con el algoritmo uuencode
- **curl_copy_handle()** - Copiar un gestor cURL junto con todas sus preferencias
- **dba_key_split()** - Separa una llave representada como cadena a una representación de matriz
- **dbase_get_header_info()** - Obtiene la información de cabecera de una base de datos dBase
- **dbx_fetch_row()** - Recupera filas desde un resultado de consulta que usó la bandera DBX_RESULT_UNBUFFERED

- **fbsql_set_password()** - Cambia la contraseña para un usuario determinado
- **file_put_contents()** - Escribe una cadena a un archivo
- **ftp_alloc()** - Reserva espacio para que el archivo sea cargado
- **get_declared_interfaces()** - Devuelve una matriz de todas las interfaces declaradas
- **get_headers()** - Recupera todas las cabeceras enviadas por el servidor en respuesta a una petición HTTP
- **headers_list()** - Devuelve una lista de cabeceras de respuesta enviadas (o listas para ser enviadas)
- **http_build_query()** - Generar una cadena query codificada estilo URL
- **image_type_to_extension()** - Obtiene la extensión de archivo para el tipo de imagen devuelto por **getimagesize()**, **exif_read_data()**, **exif_thumbnail()**, **exif_imagetype()**
- **imagefilter()** - Aplica un filtro sobre una imagen usando un ángulo personalizado
- **imap_getacl()** - Obtiene el valor ACL para la casilla de correo dada
- **ldap_sasl_bind()** - Enlaza con el directorio LDAP usando SASL
- **mb_list_encodings()** - Devuelve una matriz de todas las codificaciones soportadas
- **pcntl_getpriority()** - Obtiene la prioridad de cualquier proceso
- **pcntl_wait()** - Espera o devuelve el status de un hijo bifurcado como se define por la llamada de sistema waitpid()
- **pg_version()** - Devuelve una matriz con el cliente, protocolo y versión del servidor (cuando se encuentra disponible)
- **php_check_syntax()** - Verifica la sintaxis del archivo especificado
- **php_strip_whitespace()** - Devuelve el código fuente con los comentarios y espacios eliminados
- **proc_nice()** - Cambia la prioridad del proceso actual
- **pspell_config_data_dir()** - Cambia la ubicación de los archivos de datos del lenguaje
- **pspell_config_dict_dir()** - Cambia la ubicación de la lista principal de palabras
- **setrawcookie()** - Envía una cookie sin codificar el valor en estilo url
- **snmp_read_mib()** - Lee y realiza análisis sintáctico sobre un archivo MIB en el árbol MIB activo
- **sqlite_fetch_column_types()** - Devuelve una matriz de tipos de columna de una tabla en particular

Nota: La extensión Tidy ha modificado también su API por completo.

Nuevas Directivas

Algunas directivas de `php.ini` se han incorporado en PHP 5. Aquí se presenta una lista de ellas:

- mail.force_extra_parameters - Obliga a que se agreguen los parámetros especificados como parámetros extra al ejecutable de sendmail. Estos parámetros siempre reemplazan el valor del quinto parámetro de **mail()**, incluso en modo seguro
- register_long_arrays - habilita/deshabilita el registro de las largas variables obsoletas $HTTP_*_VARS
- session.hash_function - selecciona una función de resumen criptográfico (MD5 o SHA-1)
- session.hash_bits_per_character - define cuántos bits deben almacenarse en cada caracter cuando se convierten datos binarios de hash a algo más legible (de 4 a 6)
- zend.ze1_compatibility_mode - Habilita el modo de compatibilidad con el Motor Zend 1 (PHP 4)

Bases de Datos

Hubo algunos cambios en PHP 5 relacionados con las bases de datos (MySQL y SQLite).

En PHP 5 las bibliotecas de cliente MySQL no se incluyen, debido a problemas de licencia, entre otros. Para más información, lea la entrada del FAQ.

También hay una nueva extensión, MySQLi (MySQL Mejorado), la cual está diseñada para trabajar con MySQL 4.1 y superior.

A partir de PHP 5, la extensión SQLite viene incorporada en PHP. SQLite es un motor de bases de datos embebible y no es una biblioteca de cliente usada para conectarse con un servidor grande de bases de datos (como MySQL o PostgreSQL). La biblioteca SQLite lee y escribe directamente hacia y desde los archivos de bases de datos en disco.

Nuevo Modelo de Objetos

En PHP 5 existe un nuevo Modelo de Objetos. El manejo de objetos en PHP ha sido re-escrito por completo, permitiendo una mejora en rendimiento y muchas características nuevas. En versiones previas de PHP, los objetos eran manejados como tipos primitivos (por ejemplo enteros y cadenas). La desventaja de este método era que semánticamente el objeto completo era copiado cuando una variable era asignada, o pasada como parámetro a un método. En el nuevo enfoque, los objetos son referenciados por gestor, y no por valor (puede pensarse en el gestor como un identificador de objeto).

Muchos programadores de PHP no son conscientes siquiera de los detalles que implican las copias en el modelo antiguo de objetos y, por lo tanto, la mayoría de aplicaciones de PHP funcionarán sin

problemas, o con muy pocas modificaciones.

El nuevo Modelo de Objetos es documentado en la Referencia del Lenguaje.

Vea también la directiva zend.ze1_compatibility_mode para compatibiladad con PHP 4.

Reporte de Errores

A partir de PHP 5, la nueva constante de reporte de errores E_STRICT se encuentra disponible con el valor 2048. Ésta habilita sugerencias en tiempo de ejecución por parte de PHP sobre la inter-operabilidad de su código y compatibilidad hacia adelante, que le ayudarán a sincronizarse con los últimos y mejores métodos de escritura de código. Por ejemplo, un mensaje STRICT puede advertirle cuando use funciones obsoletas.

> **Nota:** E_ALL no incluye E_STRICT, así que esta constante no está habilitada por defecto.

Apéndice C. Migración de PHP 3 a PHP 4

Qué ha cambiado en PHP 4

PHP 4 y el motor Zend integrado han mejorado considerablemente las capacidades y el rendimiento de PHP, pero se ha tenido mucho cuidado para evitar cualquier impacto negativo sobre el código existente. De modo que migrar su código desde PHP 3 a PHP 4 debe ser mucho más sencillo que migrar de PHP/FI 2 a PHP 3. Bastante código de PHP 3 existente debe estar listo para correr sin modificaciones, pero aun así es importante que sepa sobre las pocas diferencias, y tome las precauciones necesarias para probar su código antes de cambiar las versiones en entornos de producción. Las siguientes líneas deben propocionarle algunas pistas sobre qué tipo de cosas considerar.

Ejecutar PHP 3 y PHP 4 simultáneamente

Sistemas operativos recientes ofrecen la posibilidad de realizar versionamiento y contextualización. Estas características hacen posible que PHP 3 y PHP 4 corran como módulos concurrentes en un servidor Apache.

Se conoce que esta característica funciona sobre las siguientes plataformas:

- Linux con versión reciente de binutils (se ha probado con binutils 2.9.1.0.25)
- Solaris 2.5 o superior
- FreeBSD (se ha probado con 3.2, 4.0)

Para habilitarlo, configure PHP 3 y PHP 4 para que usen APXS (*--with-apxs*) y las extensiones de enlace necesarias (*--enable-versioning*). Por lo demás, todas las instrucciones de instalación estándar se aplican. Por ejemplo:

```
$ ./configure \
  --with-apxs=/apache/bin/apxs \
  --enable-versioning \
  --with-mysql \
  --enable-track-vars
```

Migración de Archivos de Configuración

El archivo de configuración global, `php3.ini`, ha cambiado su nombre a `php.ini`.

Para el archivo de configuración de Apache, existen unos cuantos cambios más. Los tipos MIME reconocidos por el módulo PHP han cambiado.

```
application/x-httpd-php3        -->    application/x-httpd-php
application/x-httpd-php3-source -->    application/x-httpd-php-source
```

Puede hacer que sus archivos de configuración trabajen con ambas versiones de PHP (dependiendo de cuál es la que está compilada actualmente con el servidor), usando la siguiente sintaxis:

```
AddType  application/x-httpd-php3        .php3
AddType  application/x-httpd-php3-source .php3s

AddType  application/x-httpd-php         .php
AddType  application/x-httpd-php-source  .phps
```

Adicionalmente, los nombres de directivas PHP para Apache han cambiado.

A partir de PHP 4.0, existen solo cuatro directivas Apache que se relacionan con PHP:

```
php_value [nombre de directiva PHP] [valor]
php_flag [nombre de directiva PHP] [On|Off]
php_admin_value [nombre de directiva PHP] [valor]
php_admin_flag [nombre de directiva PHP] [On|Off]
```

Existen dos diferencias entre los valores Admin y los no-admin:

- Los valores (o banderas) admin pueden aparecer solo en los archivos de configuración globales de Apache (p.ej., `httpd.conf`).

- Los valores (o banderas) estándar no pueden controlar ciertas directivas PHP, por ejemplo: safe mode (si pudiera sobrescribir los parámetros del modo seguro en los archivos `.htaccess`, se estropearía la gracia de safe mode). En contraste, los valores Admin pueden modificar el valor de cualquier directiva PHP.

Para hacer el proceso de transición más sencillo, PHP 4 es distribuido con scripts que convierten automáticamente su configuración en archivos Apache y `.htaccess` para que trabajen con PHP 3 y PHP 4. ¡Estos scripts NO convierten las líneas de tipos mime! Tendrá que convertir éstas manualmente.

Para convertir sus archivos de configuración de Apache, ejecute el script apconf-conv.sh (disponible en el directorio scripts/apache/). Por ejemplo:

```
~/php4/scripts/apache:#  ./apconf-conv.sh /usr/local/apache/conf/httpd.conf
```

Su archivo de configuración original será guardado en httpd.conf.orig.

Para convertir sus archivos `.htaccess`, ejecute el script `aphtaccess-conv.sh` (disponible

así mismo en el directorio `scripts/apache/`):

```
~/php4/scripts/apache:#  find / -name .htaccess -exec ./aphtaccess-conv.sh {} \;
```

De modo semejante, sus archivos `.htaccess` antiguos serán guardados con el prefijo .orig.

Los scripts de conversión requieren que awk esté instalado.

Comportamiento del analizador sintáctico

El proceso de análisis sintáctico y la ejecución son ahora dos pasos completamente separados, no se procederá a la ejecución del código de cualquier archivo hasta que éste en su totalidad, así como todo el código requerido se haya analizado completa y satisfactoriamente.

Uno de los nuevos requisitos introducidos con esta separación es que todos los archivos requeridos y de inclusión tienen que ser sintácticamente completos ahora. Ya no es permitida la separación de diferentes segmentos de una estructura de control a través de varios archivos. Esto quiere decir que ahora no puede iniciar un ciclo *for* o *while*, una sentencia *if* o un bloque *switch* en un archivo, y tener el final del ciclo, sentencias *else*, *endif*, *case* o *break* en un archivo diferente.

Aun es perfectamente legal incluir código adicional al interior de ciclos u otras estructuras de control, únicamente las palabras claves de control y los corchetes correspondientes *{...}* tienen que estar en la misma unidad de compilación (archivo o cadena procesada por **eval()**).

Esto no debe generar una repercusión significativa ya que separar el código de esta manera debe ser considerado como muy mal estilo, en cualquier caso.

Algo más que ya no es posible, aunque es rara veces visto en código PHP 3, es devolver valores desde un archivo requerido. Devolver un valor desde un archivo de inclusión es posible aun.

Reporte de errores

Cambios de configuración

Con PHP 3 el nivel de reporte de errores estaba establecido como un valor numérico simple formado por la suma de los números relacionados con diferentes niveles de error. Algunos valores usuales eran 15 para reportar todos los errores y advertencias, o 7 para reportar todo excepto mensajes de noticias simples que indicaban mal estilo del código y cosas por el estilo.

PHP 4 tiene un conjunto de niveles de error y advertencia mayor y viene con un analizador sintáctico de configuración que permite el uso de constantes simbólicas para determinar el comportamiento deseado.

El nivel de reporte de errores debe ser ahora configurado mediante la deshabilitación explícita de niveles de advertencia que no desea que generen mensajes de error, con una sentencia lógica OR sobre la constante simbólica *E_ALL*. ¿Suena complicado? Bien, digamos que usted desea que el sistema de reporte de errores le haga saber sobre cualquier problema excepto por las advertencias de estilo simples que están categorizadas por la constante simbólica *E_NOTICE*. En ese caso, colocará lo siguiente en su `php.ini`: *error_reporting = E_ALL & ~ (E_NOTICE)*. Si desea suprimir las advertencias también, usted agrega la constante apropiada al interior de los paréntesis usando el

operador binario OR '|': *error_reporting= E_ALL & ~ (E_NOTICE | E_WARNING).*

Aviso
Cuando actualice código o servidores desde PHP 3 a PHP 4, usted debería chequear estos parámetros y llamadas a **error_reporting()** o puede que quiera deshabilitar el reporte de los nuevos tipos de error, especialmente E_COMPILE_ERROR. Esto puede llevar a la generación de documentos sin información alguna sobre qué ha pasado o en dónde investigar por posibles problemas.

Aviso
El uso de los valores antiguos 7 y 15 para establecer el reporte de errores es una muy mala idea ya que esto suprime algunas de las nuevas clases de errores presentes, incluyendo errores de la fase de análisis sintáctico. Esto puede producir comportamientos bastante extraños, debido a que posiblemente los scripts no trabajen más sin dejar de mostrar mensajes de error en todas partes. Esto ha producido una cantidad enorme de reportes de fallos irreproducibles en el pasado, en donde la gente reportaba problemas con el motor de scripts que eran incapaces de rastrear, cuando lo que sucedía en realidad era usualmente algún '}' que faltaba en un archivo requerido, que el analizador sintáctico no podía reportar debido a un sistema de reporte de errores mal configurado. Así que revisar la configuración de su reporte de errores debe ser lo primero que debe hacer siempre que sus scripts mueran silenciosamente. El motor Zend puede considerarse suficientemente maduro en la actualidad como para afirmar que éste no es el causante de estos comportamientos extraños.

Mensajes de advertencia adicionales

Una gran cantidad de código PHP 3 existente usa construcciones del lenguaje que deben ser consideradas como muy mal estilo ya que, aunque logran el efecto esperado ahora, pueden verse influenciadas por cambios en otros lugares del código. PHP 4 desplegará una enorme cantidad de mensajes de noticia en tales situaciones en donde PHP 3 no lo hacía. La solución simple es deshabilitar los mensajes E_NOTICE, pero usualmente es una buena idea arreglar el código en su lugar.

El caso más común que produce mensajes de noticia ahora es el uso de constantes de cadena sin comillas como índices de matrices. Tanto PHP 3 como PHP 4 llegarán a interpretar éstas como cadenas si no existen palabras clave o constantes con tales nombres, pero en donde sea que una constante con tal nombre haya sido definida en algún lugar del código, puede dañar el script. Esto puede convertirse incluso en un riesgo de seguridad si algún intruso logra redefinir constantes de cadena en forma tal que hace que sus scripts le den derechos de acceso que él no debía tener. Así que PHP 4 le advertirá ahora siempre que use constantes de cadena sin comillas, como por ejemplo en *$_SERVER[REQUEST_METHOD]*. Modificar tal expresión por *$_SERVER ['REQUEST_METHOD']* hará feliz al analizador sintáctico y mejorará significativamente el estilo y la seguridad de su código.

Otra cosa sobre la que PHP 4 le notificará es sobre el uso de variables o elementos de matrices sin inicializar.

Inicializadores

Las variables estáticas y los inicializadores de miembros de clase aceptan únicamente valores

escalares, mientras que en PHP 3 aceptaban cualquier expresión válida. Esto es, nuevamente, debido a la separación entre el análisis sintáctico y la ejecución ya que el código no ha sido ejecutado aun cuando el analizador sintáctico ve el inicializador.

Para clases, debería usar constructores para inicializar variables miembro en su lugar. Para variables estáticas, cosas diferentes a simples valores estáticos rara vez tienen sentido después de todo.

---

empty("0")

Quizás el cambio de comportamiento más controversial ha ocurrido con el modo en que trabaja **empty()**. Una cadena que contenga solo el caracter '0' (cero) es considerada vacía, mientras que en PHP 3 no era así.

Este nuevo comportamiento tiene sentido en aplicaciones web, dado que todos los campos de entrada devuelven cadenas incluso si se solicitan valores numéricos, y dadas las capacidades de conversión automática de tipos de PHP. Pero, por otra parte, puede dañar su código en formas sutiles, causando comportamientos exóticos que son difíciles de rastrear si no sabe qué buscar.

Funciones faltantes

Aunque PHP 4 viene con una gran cantidad de características, funciones y extensiones nuevas, puede que aun encuentre funciones de la versión 3 que hacen falta. Un número pequeño de funciones centrales han desaparecido ya que no funcionan con el nuevo esquema de separación de análisis sintáctico y ejecución que se introdujo en PHP 4 con el motor Zend. Otras funciones, e incluso extensiones completas, se han marcado obsoletas a medida que nuevas funciones y extensiones cubren las mismas tareas y en ocasiones en una forma más general. Algunas funciones simplemente no han sido portadas aun y finalmente algunas funciones o extensiones pueden faltar debido a conflictos de licencias.

Funciones faltantes debido a cambios conceptuales

Dado que PHP 4 separa ahora el análisis sintáctico de la ejecución, ya no es posible modificar el comportamiente del analizador sintáctico (embebido ahora en el motor Zend) en tiempo de ejecución, dado que el análisis ya ha ocurrido para entonces. De modo que la función **short_tags()** ya no existe. Aun puede modificar el comportamiento del analizador sintánctico definiendo los valores apropiados en el archivo `php.ini`.

Otra característica de PHP 3 que no es parte de PHP 4 es la interfaz de depuración integrada. Existen adiciones de terceros para el motor Zend que añaden funcionalidades similares.

Funciones y extensiones deprecadas

Las extensiones de bases de datos Adabas y Solid ya no existen. Larga vida a la extensión unificada ODBC en su lugar.

Status modificado para unset()

unset(), aun cuando sigue estando disponible, es implementada ahora como una construcción del lenguaje en lugar de una función.

Esto no tiene consecuencia alguna en el comportamiento de **unset()**, pero realizar una prueba con "unset" usando **function_exists()** devolverá **FALSE** del mismo modo que ocurriría con otras construcciones del lenguaje que tienen apariencia de funciones, como **echo()**.

Otro cambio más práctico es que ya no es posible llamar **unset()** indirectamente, esto es, *$func="unset"; $func($alguna_variable)* no funcionará más.

Extensiones PHP 3

Las extensiones escritas para PHP 3 no trabajarán con PHP 4, ni como binarios ni al nivel de fuente. No es difícil portar extensiones a PHP 4 si tiene acceso a las fuentes originales. Una descripción detallada de tal proceso de migración no hace parte de este documento.

Sustitución de variables en cadenas

PHP 4 añade un nuevo mecanismo para la sustitución de variables en cadenas. Ahora, finalmente, puede acceder a variables miembro de objetos y elementos de matrices multidimensionales al interior de cadenas.

Para tal efecto, necesita rodear sus variables con corchetes, colocando el signo de dólar inmediatamente después del corchete de apertura: *{$...}*

Para embeber el valor de una variable miembro de objeto en una cadena, simplemente escriba *"texto {$obj->miembro} texto"*, mientras que en PHP 3 debía hacer algo como *"texto ".$obj->miembro." texto"*.

Esto debería representar código más legible, aunque podría arruinar scripts existentes escritos para PHP 3. Pero puede encontrar fácilmente este tipo de problemas, revisando el código por la combinación de caracteres *{$* en su código, y reemplazándola por *\{$* con su herramienta de búsqueda-y-reemplazo favorita.

Cookies

PHP 3 tenía el mal hábito de definir cookies en el orden contrario al de las llamadas a **setcookie()** en su código. PHP 4 acabó con este hábito y crea las líneas de cabecera de cookies en el mismo orden exacto en el que usted define las cookies en el código.

Esto puede dañar código existente, pero el comportamiento antiguo era tan extraño de entender que merecía un cambio para prevenir posteriores problemas en el futuro.

Gestión de variables globales

Aunque el manejo de variables globales se enfocaba en la simplicidad en PHP 3 y versiones tempranas de PHP 4, éste enfoque ha cambiado para ser más seguro. Mientras que en PHP 3 el siguiente ejemplo funcionaba bien, en PHP 4 debe ser unset(*unset($GLOBALS["id"])*);. Este es sólo un detalle de la gestión de variables globales. Usted debería usar siempre *$GLOBALS*, con versiones recientes de PHP 4 está obligado a hacerlo en la mayoría de casos. Lea más sobre este asunto en la sección de referencias *global*.

Ejemplo C-1. Migración de variables globales

```
<?php
$id = 1;
function prueba()
{
    global $id;
    unset($id);
}
prueba();
echo($id); // Esto imprime 1 en PHP 4
?>
```

Apéndice D. Migración desde PHP/FI 2 hacia PHP 3

Sobre las incompatibilidades en 3.0

PHP 3.0 ha sido rescrito desde ceros. Posee un analizador sintáctico apropiado, que es mucho más robusto y consistente que el de 2.0. 3.0 es también significativamente mas rápido, y usa menos memoria. Sin embargo, algunas de estas mejoras no fueron posibles sin modificaciones de compatibilidad, tanto en la sintaxis como en el funcionamiento.

Además, los desarrolladores de PHP han intentado limpiar tanto la sintaxis como la semántica de PHP en la versión 3.0, y ésto ha causado también algunas incompatibilidades. Creemos que, a largo plazo, estos cambios serán para bien.

Este capítulo intentará guiarle a través de las incompatibilidades que puede encontrar cuando vaya desde PHP/FI 2.0 a PHP 3.0, además de ayudarle a resolverlas. No se mencionan aquí las nuevas características, a menos que sea necesario.

Existe un programa que puede convertir automáticamente sus viejos scripts PHP/FI 2.0. Puede ser encontrado en el subdirectorio `convertor` de la distribucion 3.0 de PHP. Sin embargo, éste programa solo detecta los cambios de sintaxis, así que debería leer este capítulo con cuidado, en cualquier caso.

---

old_function

La sentencia *old_function* le permite declarar una función usando una sintaxis idéntica a PHP/FI2 (excepto en que debe reemplazar 'function' con 'old_function').

Esta es una característica considerada como obsoleta, y debe ser usada únicamente por el programa

de conversión PHP/FI2->PHP 3.

Aviso
Las funciones declaradas como *old_function* no pueden ser llamadas desde el código interno de PHP. Entre otras cosas, esto quiere decir que no puede usarlas en funciones como **usort()**, **array_walk()**, y **register_shutdown_function()**. Puede evitar esta limitación escribiendo una función de envoltura (en la forma normal de PHP 3) para llamar a *old_function*.

Etiquetas de inicio/final

Aquello que probablemente note primero, es que las etiquetas de inicio y final de PHP han cambiado. Las vieja forma <? > ha sido reemplazada por tres nuevas formas posibles:

Ejemplo D-1. Migración: etiquetas viejas de inicio/fin

```
<? echo "Esto es c&oacute;digo PHP/FI 2.0.\n"; >
```

A partir de la versión 2.0, PHP/FI también soporta esta variante:

Ejemplo D-2. Migración: primer juego nuevo de etiquetas de inicio/fin

```
<? echo "&iexcl;Esto es c&oacute;digo PHP 3.0!\n"; ?>
```

Note que la etiqueta de fin consiste ahora en un signo de interrogación y un caracter mayor-que, en lugar de tan sólo un mayor-que. Sin embargo, si planea usar XML en su servidor, tendrá problemas con la primera variante nueva, ya que PHP puede intentar ejecutar el marcado XML en documentos XML como código PHP. Es por esto que se han introducido las siguientes variantes:

Ejemplo D-3. Migración: segundo juego de etiquetas de inicio/fin

```
<?php echo "&iexcl;Esto es c&oacute;digo PHP 3.0!\n"; ?>
```

Algunas personas han tenido problemas con editores que no reconocen las etiquetas de instrucción de procesamiento después de todo. Microsoft FrontPage es uno de esos editores, y a modo de remedio, se ha introducido también esta otra variante:

Ejemplo D-4. Migración: tercer juego de etiquetas de inicio/fin

```
<script language="php">

  echo "&iexcl;Esto es c&oacute;digo PHP 3.0!\n";

</script>
```

Sintaxis if..endif

El modo `alternativo' de escribir sentencias if/elseif/else, usando if();elseif(); else; endif; no puede implementarse de forma eficiente sin agregar una gran cantidad de complejidad al analizador sintáctico 3.0. Debido a esto, la sintaxis ha sido modificada:

Ejemplo D-5. Migración: sintaxis if..endif antigua

```
if ($foo);
    echo "sip\n";
elseif ($bar);
    echo "casi\n";
else;
    echo "nop\n";
endif;
```

Ejemplo D-6. Migración: sintaxis if..endif nueva

```
if ($foo):
    echo "sip\n";
elseif ($bar):
    echo "casi\n";
else:
    echo "nop\n";
endif;
```

Note que los punto-y-coma, han sido reemplazados por los los dos-puntos en todas las sentencias, excepto en aquella que termina la expresión (endif).

Sintaxis while

Al igual que con if..endif, la sintaxis de while..endwhile ha cambiado también:

Ejemplo D-7. Migración: sintaxis while..endwhile antigua

```
while ($mas_en_camino);
    ...
endwhile;
```

Ejemplo D-8. Migración: sintaxis while..endwhile nueva

```
while ($mas_en_camino):
    ...
endwhile;
```

Aviso
Si usa la sintaxis antigua while..endwhile en PHP 3.0, obtendrá un ciclo que nunca finaliza.

Tipos de las expresiones

PHP/FI 2.0 usaba el lado izquierdo de las expresiones para determinar el tipo del resultado. PHP 3.0 toma ambos lados en cuenta cuando determina los tipos del resultado, y esto puede producir comportamientos impredecibles si ejecuta scripts 2.0 en 3.0.

Considere este ejemplo:

```
$a[0]=5;
$a[1]=7;

$clave = key($a);
while ("" != $clave) {
    echo "$clave";
    next($a);
}
```

En PHP/FI 2.0, esto mostraría los dos índices de $a. En PHP 3.0, no mostraría nada. La razón es que en PHP 2.0, al ser el argumento izquierdo de tipo cadena, se realizaba una comparación de cadenas, y ciertamente *""* no es igual a *"0"*, y el ciclo continúa. En PHP 3.0, cuando se compara una cadena con un entero, se realiza una comparación de enteros (la cadena es convertida a entero). Esto resulta en la comparación de *atoi("")* que es *0*, y *lista_de_variables* que también es *0*, y dado que *0==0*, el ciclo no avanzará ni una sola vez.

La solución a esto es simple. Reemplace la sentencia while con:

```
while ((string)$clave != "") {
```

Los mensajes de error han cambiado

Los mensajes de error de PHP 3.0 son, usualmente, más precisos que los de 2.0, pero ya no puede ver el fragmento de código que causa el error. Sin embargo, se le entregará el nombre de archivo y número de línea del error.

Evaluación booleana de corto-circuito

En PHP 3.0, la evaluación booleana es por corto-circuito. Esto significa que en una expresión como *(1 || prueba())*, la función **prueba()** no será ejecutada ya que nada puede cambiar el resultado de la expresión después del *1*.

Es éste un detalle menor de compatibilidad, pero puede provocar inesperados efectos colaterales.

Valores de retorno de función `TRUE`/`FALSE`

La mayoría de funciones internas han sido rescritas de modo que devuelvan **`TRUE`** de tener éxito y **`FALSE`** cuando fallan, en contraste a los valores 0 y -1 de PHP/FI 2.0, respectivamente. El nuevo comportamiente permite la implementación de código más lógico, como *$da = fopen("/su/archivo") or fail("¡diablos!");*. Dado que PHP/FI 2.0 no disponía de reglas claras sobre el valor que debían devolver las funciones cuando fallan, la mayoría de tales scripts probablemente deban revisarse manualmente después de usar el programa de conversión de 2.0 a 3.0.

Ejemplo D-9. Migración desde 2.0: valores de retorno, código antiguo

```
$da = fopen($archivo, "r");
if ($da == -1);
    echo("No pudo abrirse $archivo para lectura<br />\n");
endif;
```

Ejemplo D-10. Migración desde 2.0: valores de retorno, código nuevo

```
$da = @fopen($archivo, "r") or print("No pudo abrirse $archivo para lectura<br />\n");
```

Otras incompatibilidades

- El módulo Apache de PHP 3.0 no soporta más versiones de Apache anteriores a la 1.2. Se requiere Apache 1.2 o superior.
- **echo()** no soporta más una cadena de formato. Use la función **printf()** en su lugar.
- En PHP/FI 2.0, un efecto colateral de implementación causaba que *$foo[0]* tuviera el mismo efecto que *$foo*. Esto no ocurre en PHP 3.0.
- La lectura de matrices con *$matriz[]* ya no es soportada.

  Esto quiere decir que no puede recorrer una matriz mediante un ciclo que realice *$datos = $matriz[]*. Use **current()** y **next()** en su lugar.

  Así mismo, *$matriz1[] = $matriz2* no adiciona los valores de *$matriz2* a *$matriz1*, sino que adiciona *$matriz2* como la última entrada de *$matriz1*. Consulte también el soporte de

matrices multidimensionales.

- ″+″ ya no es sobrecargado como un operador de concatenación de cadenas, sino que convierte sus argumentos a números y realiza una suma numérica. Use ″.″ en su lugar.

Ejemplo D-11. Migración desde 2.0: concatenación para cadenas

```
echo "1" + "1";
```

En PHP 2.0 esto mostraría 11, en PHP 3.0 imprimiría 2. En su lugar use:

```
echo "1"."1";

$a = 1;
$b = 1;
echo $a + $b;
```

Esto imprimiría 2 tanto en PHP 2.0 como en 3.0.

```
$a = 1;
$b = 1;
echo $a.$b;
```

Esto imprimirá 11 en PHP 3.0.

Apéndice E. Depuración en PHP

Sobre el depurador

PHP 3 incluye soporte para un depurador basado en red.

PHP 4 no cuenta con una herramienta de depuración interna. Aun así, puede usar uno de los depuradores externos. El entorno integrado de desarrollo Zend incluye un depurador, y existen también algunas extensiones de depuración gratuitas como DBG en http://dd.cron.ru/dbg/, el Depurador Avanzado de PHP (APD, por sus siglas en Inglés) o Xdebug, el cual posee incluso una interfaz de depuración compatible con la funcionalidad de depuración de PHP 3, tal y como se describe en esta sección.

Uso del Depurador

El depurador interno en PHP 3 es útil para rastrear fallos difíciles de señalar. El depurador funciona creando una conexión con un puerto TCP en cada ocasión que PHP 3 arranca. Todos los mensajes de error de esa petición serán enviados a esta conexión TCP. Esta información tiene como propósito ser usada por un "servidor de depuración" que puede correr dentro de un IDE o editor programable (como Emacs).

Cómo configurar el depurador:

1. Configure un puerto TCP para el depurador en el archivo de configuración (debugger.port) y habilítelo (debugger.enabled).

2. Configure un receptor TCP en ese puerto en alguna parte (por ejemplo, **socket -l -s 1400** en sistemas Unix).

3. En su código, ejecute "debugger_on(*host*)", en donde *host* es el número IP o nombre del servidor huésped en donde está corriendo el receptor TCP.

Ahora, todas las advertencias, noticias etc. aparecerán en ese socket receptor, *incluso si las ha deshabilitado con* ***error_reporting()***.

Protocolo del Depurador

El protocolo del depurador de PHP 3 es basado en líneas. Cada línea tiene un *tipo*, y varias líneas componen un *mensaje*. Cada mensaje comienza con una línea del tipo *start* y termina con una línea del tipo *end*. PHP 3 puede enviar líneas para diferentes mensajes simultáneamente.

Una línea tiene este formato:

```
fecha hora host(pid) tipo: datos-mensaje
```

fecha

Fecha en formato ISO 8601 (*aaaa-mm-dd*)

hora

Hora, incluyendo microsegundos: *hh*:*mm*:*uuuuuu*

host

Nombre DNS o dirección IP del host en donde se generó el error de script.

pid

PID (id de proceso) en *host* del proceso que contenía el script de PHP 3 que generó este error.

tipo

Tipo de línea. Le dice al programa receptor sobre el modo en que debería tratar los datos a continuación:

Tabla E-1. Tipos de Línea del Depurador

Nombre	Significado
start	Le dice el programa receptor que un mensaje de depurador comienza aquí. Los contenidos de *datos* serán el tipo de mensaje de error, listados más adelante.
message	El mensaje de error PHP 3.
location	Nombre de archivo y número de línea en donde ocurrió el error. La primera línea *location* contendrá siempre la ubicación de nivel más alto. *datos* contendrá *archivo:línea*.Siempre habrá una línea *location* después de *message* y después de cada *function*.
frames	Número de marcos en el siguiente volcado de pila. Si hay cuatro marcos, espere información sobre cuatro niveles de llamados a función. Si no se entrega una línea "frames", se asumirá que la profundidad es 0 (el error ocurrió en el contexto de más alto nivel).
function	Nombre de la función en donde ocurrió el error. Será repetido una vez por cada nivel en la pila de llamados a funciones.

Nombre	Significado
end	Le indica al programa receptor que el mensaje de depurador termina aquí.

datos

Datos de la línea.

Tabla E-2. Tipos de Error del Depurador

Depurador	Interno de PHP 3
warning	E_WARNING
error	E_ERROR
parse	E_PARSE
notice	E_NOTICE
core-error	E_CORE_ERROR
core-warning	E_CORE_WARNING
unknown	(cualquier otro)

Ejemplo E-1. Mensaje de Depurador de Ejemplo

```
1998-04-05 23:27:400966 lucifer.guardian.no(20481) start: notice
1998-04-05 23:27:400966 lucifer.guardian.no(20481) message: Uninitialized variable
1998-04-05 23:27:400966 lucifer.guardian.no(20481) location: (null):7
1998-04-05 23:27:400966 lucifer.guardian.no(20481) frames: 1
1998-04-05 23:27:400966 lucifer.guardian.no(20481) function: display
1998-04-05 23:27:400966 lucifer.guardian.no(20481) location: /home/ssb/public_html/test.php3:10
1998-04-05 23:27:400966 lucifer.guardian.no(20481) end: notice
```

Apéndice F. Extensión de PHP 3

Esta sección se encuentra bastante desactualizada y demuestra el modo de extender PHP 3. Si está interesado en PHP 4, por favor lea la sección sobre la interfaz de programación Zend. Así mismo, usted querrá leer varios archivos encontrados en el código fuente de PHP, archivos como `README.SELF-CONTAINED-EXTENSIONS` y `README.EXT_SKEL`.

Adición de funciones a PHP

Prototipo de Función

Todas las funciones lucen de este modo:

```
void php3_foo(INTERNAL_FUNCTION_PARAMETERS) {

}
```

Incluso si su función no recibe argumentos, ésta es la forma de llamarla.

Argumentos de Función

Los argumentos son siempre de tipo pval. Este tipo posee una unión que contiene el tipo real del argumento. Así que, si su función recibe dos argumentos, usted haría algo como lo siguiente al comienzo de su función:

Ejemplo F-1. Recuperación de argumentos de función

```
pval *arg1, *arg2;
if (ARG_COUNT(ht) != 2 || getParameters(ht,2,&arg1,&arg2)==FAILURE) {
   WRONG_PARAM_COUNT;
}
```

NOTA: Los argumentos pueden ser pasados ya sea por valor o por referencia. En ambos casos necesitará pasar &(pval *) a getParameters. Si desea chequear si el parámetra n'ésimo le fue enviado por referencia o no, puede usar la función ParameterPassedByReference(ht,n). Ésta devolverá 1 o 0.

Cuando usted modifica cualquiera de los parámetros pasados, ya sea que hayan sido enviados por referencia o por valor, puede o bien comenzar con el parámetro llamando pval_destructor sobre él, o si es un ARRAY al que desea agregar valores, puede usar funciones similares a aquellas en internal_functions.h que manipulan return_value como un ARRAY.

También, si modifica un parámetro a IS_STRING asegúrese de asignar primero la nueva cadena mediante estrdup() y la longitud de la cadena, y sólo después modifique el tipo a IS_STRING. Si modifica la cadena de un parámetro que ya es IS_STRING o IS_ARRAY, debe ejecutar pval_destructor sobre éste primero.

Argumentos de Función Variables

Una función puede recibir un número variable de argumentos. Si su función puede recibir ya sea 2 o 3 argumentos, use lo siguiente:

Ejemplo F-2. Argumentos de función variables

```
pval *arg1, *arg2, *arg3;
int arg_count = ARG_COUNT(ht);

if (arg_count < 2 || arg_count > 3 ||
    getParameters(ht,arg_count,&arg1,&arg2,&arg3)==FAILURE) {
    WRONG_PARAM_COUNT;
}
```

Uso de los Argumentos de Función

El tipo de cada argumento es almacenado en el campo type de pval. Este tipo puede ser cualquiera de los siguientes:

Tabla F-1. Tipos Internos de PHP

IS_STRING	Cadena
IS_DOUBLE	Punto flotante de doble precisión

IS_LONG	Entero largo
IS_ARRAY	Matriz
IS_EMPTY	Ninguno
IS_USER_FUNCTION	??
IS_INTERNAL_FUNCTION	?? (si alguno de éstos no puede ser pasado a una función - eliminar)
IS_CLASS	??
IS_OBJECT	??

Si recibe un argumento de un tipo y quisiera usarlo como otro, o si tan sólo desea obligar al argumento a que sea de un cierto tipo, puede usar una de las siguientes funciones de conversión:

```
convert_to_long(arg1);
convert_to_double(arg1);
convert_to_string(arg1);
convert_to_boolean_long(arg1); /* Si la cadena es "" o "0" se convierte a 0, 1 de lo co
convert_string_to_number(arg1);  /* Convierte una cadena a un LONG o DOUBLE, dependiend
```

Estas funciones todas realizan conversión en-el-lugar. No devuelven nada.

El argumento como tal es almacenado en una unión; los miembros son:

- IS_STRING: arg1->value.str.val
- IS_LONG: arg1->value.lval
- IS_DOUBLE: arg1->value.dval

Administración de Memoria en las Funciones

Cualquier segmento de memoria necesitado por una función debe ser reservado ya sea con emalloc() o estrdup(). Estas son funciones que abstraen la gestión de memoria y lucen y huelen como las funciones normales malloc() y strdup(). La memoria debe ser liberada con efree().

Hay dos tipos de memoria en este programa: la memoria que es devuelta al intérprete en una variable, y la memoria que necesita para el almacenamiento temporal en su función interna. Cuando asigna una cadena a una variable que es devuelta al intérprete, necesita asegurarse de reservar primero la memoria con emalloc() o estrdup(). Esta memoria no debería ser liberada por usted NUNCA, a menos que más adelante en la misma función sobrescriba su asignación original (aunque este tipo de práctica no se considera apropiada).

Para cualquier segmento de memoria temporal/permanente que necesite en sus funciones/bibliotecas, usted debería usar las tres funciones emalloc(), estrdup(), y efree(). Éstas se comportan EXACTAMENTE como sus contrapartes. Cualquier cosa que reserve con emalloc() o estrdup() debe liberarla con efree() en alguno u otro punto, a menos que espere que permanezca hasta el final del programa; de otro modo, habrá una fuga de memoria. El significado de "las funciones se comportan exactamente como sus contrapartes" es: si usted usa efree() sobre algo que no fue reservado con emalloc() ni estrdup(), puede que reciba un fallo de segmentación. De modo que, por favor, tenga cuidado y libere toda su memoria desperdiciada.

Si compila con "-DDEBUG", PHP imprimirá una lista de toda la memoria que fue reservada usando emalloc() y estrdup() y nunca liberada con efree() una vez termina la ejecución el script especificado.

Definición de Variables en la Tabla de Símbolos

Un número de macros se encuentra a su disposición para facilitar la definición de una variable en la tabla de símbolos:

- SET_VAR_STRING(nombre,valor)
- SET_VAR_DOUBLE(nombre,valor)
- SET_VAR_LONG(nombre,valor)

Aviso
Tenga cuidado con SET_VAR_STRING. La parte del valor debe ser reservada manualmente con malloc, ya que el código de gestión de memoria intentará liberar este apuntador más adelante. No pase memoria reservada estáticamente a un llamado a SET_VAR_STRING.

Las tablas de símbolos en PHP se encuentran implementadas como tablas asociativas. En cualquier momento dado, &symbol_table es un apuntador a la tabla de símbolos 'principal', y active_symbol_table apunta a la tabla de símbolos activa actualmente (éstas pueden ser idénticas, como al arranque, o diferentes, si se encuentra en el interior de una función).

Los siguientes ejemplos usan 'active_symbol_table'. Debe reemplazar este valor con &symbol_table si desea trabajar específicamente con la tabla de símbolos 'principal'. Asimismo, las mismas funciones pueden ser aplicadas sobre matrices, como se explica más adelante.

Ejemplo F-3. Chequear si *$foo* existe en la tabla de símbolos

```
if (hash_exists(active_symbol_table,"foo",sizeof("foo"))) { existe... }
else { no existe }
```

Ejemplo F-4. Encontrar el tamaño de una variable en una tabla de símbolos

```
hash_find(active_symbol_table,"foo",sizeof("foo"),&pvalue);
check(pvalue.type);
```

Las matrices en PHP son implementadas usando las mismas tablas asociativas como tablas de símbolos. Esto quiere decir que las dos funciones anteriores pueden ser usadas también para chequear variables al interior de matrices.

Si desea definir una nueva matriz en una tabla de símbolos, debe hacer lo siguiente.

Primero, puede que desee chequear si existe, y abortar la ejecución apropiadamente, usando hash_exists() o hash_find().

A continuación, inicialice la matriz:

Ejemplo F-5. Inicialización de una nueva matriz

```
pval arr;

if (array_init(&arr) == FAILURE) { fallo... };
hash_update(active_symbol_table,"foo",sizeof("foo"),&arr,sizeof(pval),NULL);
```

Este código declara una nueva matriz, llamada *$foo*, en la tabla de símbolos actual. Esta matriz se encuentra vacía.

Este es el modo de agregar entradas en ella:

Ejemplo F-6. Adición de nuevas entradas en una matriz nueva

```
pval entrada;

entrada.type = IS_LONG;
entrada.value.lval = 5;

/* define $foo["bar"] = 5 */
hash_update(arr.value.ht,"bar",sizeof("bar"),&entrada,sizeof(pval),NULL);

/* define $foo[7] = 5 */
hash_index_update(arr.value.ht,7,&entrada,sizeof(pval),NULL);

/* define el siguiente lugar libre en $foo[],
 * $foo[8], como 5 (funciona como en php2)
 */
hash_next_index_insert(arr.value.ht,&entrada,sizeof(pval),NULL);
```

Si desea modificar un valor que ha insertado a una matriz asociativa, primero debe recuperarlo desde la matriz. Para prevenir ineficiencias, puede ofrecer un pval ** a la función de adición de la matriz asociativa, y éste será actualizado con la dirección pval * del elemento insertado en la matriz. Si tal valor es **`NULL`** (como en todos los ejemplos anteriores) - el parámetro es ignorado.

hash_next_index_insert() usa más o menos la misma lógica que *$foo[] = bar;* en PHP 2.0.

Si está construyendo una matriz para devolverla desde una función, puede inicializar la matriz tal y como se ha indicado, haciendo:

```
if (array_init(return_value) == FAILURE) { fallo...; }
```

...y luego agregar valores con las funciones de ayuda:

```
add_next_index_long(valor_retorno,valor_long);
add_next_index_double(valor_retorno,valor_double);
add_next_index_string(valor_retorno,estrdup(valor_string));
```

Por supuesto, si la adición no es realizada correctamente luego de la inicialización de la matriz, probablemente tenga que verificar la existencia de la matriz primero:

```
pval *arr;

if (hash_find(active_symbol_table,"foo",sizeof("foo"),(void **)&arr)==FAILURE) { no se
else { use arr->value.ht... }
```

Note que hash_find recibe un apuntador a un apuntador pval, y no un apuntador pval.

Prácticamente toda función de matriz asociativa devuelve SUCCESS o FAILURE (excepto por hash_exists(), que devuelve un valor booleano de verdad).

Devolución de valores simples

Un número de macros se encuentra a su disposición para facilitar la devolución de valores desde una función.

Las macros RETURN_* todas establecen el valor de retorno y generan una devolución desde la función:

- RETURN
- RETURN_FALSE

- RETURN_TRUE
- RETURN_LONG(l)
- RETURN_STRING(s,dup) Si dup es **TRUE**, duplica la cadena
- RETURN_STRINGL(s,l,dup) Devuelve una cadena (s) especificando la longitud (l).
- RETURN_DOUBLE(d)

Las macros RETVAL_* establecen el valor de retorno, pero no devuelven.

- RETVAL_FALSE
- RETVAL_TRUE
- RETVAL_LONG(l)
- RETVAL_STRING(s,dup) Si dup es **TRUE**, duplica la cadena
- RETVAL_STRINGL(s,l,dup) Devuelve una cadena (s) especificando la longitud (l).
- RETVAL_DOUBLE(d)

Todas las macros de cadena anteriores aplicarán estrdup() sobre el argumento 's' pasado, de modo que puede liberar de forma segura el argumento después de llamar la macro, o alternativamente puede usar memoria reservada estáticamente.

Si su función devuelve repuesteas booleanas de éxito/error, use siempre RETURN_TRUE y RETURN_FALSE respectivamente.

Devolución de valores complejos

Su función también puede devolver un tipo de datos complejo como un objeto o una matriz.

Devolución de un objeto:

1. Llame object_init(valor_retorno).
2. Llénelo con valores. Las funciones disponibles para este propósito se listan más adelante.
3. Posiblemente, registre funciones para este objeto. Para obtener valores del objeto, la función tendría que recuperar "this" desde active_symbol_table. Su tipo debe ser IS_OBJECT, y es básicamente una tabla asociativa regular (esto quiere decir, puede usar las funciones de matrices asociativas regulares sobre .value.ht). El registro como tal de la función puede realizarse usando:
   `add_method( valor_retorno, nombre_funcion, apuntador_funcion );`

Las funciones usadas para poblar el objeto son:

- add_property_long(valor_retorno, nombre_propiedad, l) - Agregar una propiedad llamada 'nombre_propiedad', de tipo long, igual a 'l'
- add_property_double(valor_retorno, nombre_propiedad, d) - Igual, sólo agrega un double

- add_property_string(valor_retorno, nombre_propiedad, str) - Igual, sólo agrega una cadena
- add_property_stringl(valor_retorno, nombre_propiedad, str, l) - Igual, sólo agrega una cadena de longitud 'l'

Devolución de una matriz:

1. Llame array_init(valor_retorno).
2. Llénela con valores. Las funciones disponibles para este propósito se listan más adelante.

Las funciones usadas para poblar una matriz son:

- add_assoc_long(valor_retorno,clave,l) - agregar una matriz asociativa con la clave 'clave' y el valor largo 'l'
- add_assoc_double(valor_retorno,clave,d)
- add_assoc_string(valor_retorno,clave,cadena,duplicar)
- add_assoc_stringl(valor_retorno,clave,cadena,longitud,duplicar) especifica la longitud de la cadena
- add_index_long(valor_retorno,indice,l) - agregar una entrada en el indice 'indice' con el valor long 'l'
- add_index_double(valor_retorno,indice,d)
- add_index_string(valor_retorno,indice,cadena)
- add_index_stringl(valor_retorno,indice,cadena,length) - especificar la longitud de la cadena
- add_next_index_long(valor_retorno,l) - agregar una entrada de la matriz en la siguiente ubicación libre con el valor long 'l'
- add_next_index_double(valor_retorno,d)
- add_next_index_string(valor_retorno,cadena)
- add_next_index_stringl(valor_retorno,cadena,length) - especificar la longitud de la cadena

---

Uso de la lista de recursos

PHP posee una forma estándar de tratar con los varios tipos de recursos. Esto reemplaza todas las listas enlazadas locales usadas en PHP 2.0.

Funciones disponibles:

- php3_list_insert(apuntador, tipo) - devuelve el 'id' del recurso recién insertado
- php3_list_delete(id) - eliminar el recurso con el id especificado

- php3_list_find(id,*tipo) - devuelve el apuntador del recurso con el id especificado, actualiza 'tipo' al tipo del recurso

Típicamente, estas funciones son usadas para gestores de SQL, pero pueden ser usadas para cualquier otra cosa; por ejemplo, mantener descriptores de archivo.

Un listado de código típico luciría de la siguiente forma:

Ejemplo F-7. Adición de un nuevo recurso

```
RESOURCE *recurso;

/* ...reservar memoria para el recurso y adquirirlo... */
/* agregar un nuevo recurso a la lista */
valor_retorno->value.lval = php3_list_insert((void *) recurso, LE_RESOURCE_TYPE);
valor_retorno->type = IS_LONG;
```

Ejemplo F-8. Uso de un recurso existente

```
pval *id_recurso;
RESOURCE *recurso;
int tipo;

convert_to_long(id_recurso);
recurso = php3_list_find(id_recurso->value.lval, &tipo);
if (tipo != LE_RESOURCE_TYPE) {
    php3_error(E_WARNING,"el indice de recurso %d tiene el tipo ñocado",id_recurso->val
    RETURN_FALSE;
}
/* ...usar el recurso... */
```

Ejemplo F-9. Eliminar un recurso existente

```
pval *id_recurso;
RESOURCE *recurso;
int tipo;

convert_to_long(id_recurso);
php3_list_delete(id_recurso->value.lval);
```

Los tipos de recurso deberían estar registrados en php3_list.h, en enum list_entry_type. Adicionalmente, debe procurarse la implementación de código de finalización para cada nuevo tipo de recurso definido, en list_entry_destructor() ubicado en list.c (incluso si no tiene nada que hacer en la finalización, debe agregar un caso vacío).

Uso de la tabla de recursos persistentes

PHP posee una forma estándar de almacenar recursos persistentes (es decir, recursos que son conservados entre peticiones). El primer módulo en usar esta característica fue el módulo MySQL, y mSQL a continuación, de modo que puede obtener una idea general de cómo debe ser usado un recurso persistente leyendo mysql.c. Las funciones que debe consultar son:

php3_mysql_do_connect
php3_mysql_connect()
php3_mysql_pconnect()
La idea general de los módulos de persistencia es la siguiente:

1. Escriba todo el código de su módulo para que trabaje con la lista de recursos normales mencionada en la sección (9).

2. Escriba el código de funciones de conexión extra que revisen si el recurso ya existe en la lista de

recursos persistentes. Si es así, regístrelo en la lista de recursos normal como un apuntador a la lista de recursos persistentes (debido a 1., el resto del código debe funcionar inmediatamente). Si no existe, entonces créelo, agréguelo a la lista de recursos persistentes Y agregue un apuntador hacia él desde la lista normal de recursos, de modo que todo el código pueda funcionar; esto ya que se encuentra en la lista de recursos regulares, pero, en la siguiente conexión, el recurso sería encontrado en la lista de recursos persistentes y usado sin tener que crearlo de nuevo. Debe registrar éstos recursos con un tipo diferente (p.ej. LE_MYSQL_LINK para un enlace no-persistente y LE_MYSQL_PLINK para un enlace persistente).

Si lee mysql.c, notará que, con la excepción de la función de conexión más compleja, nada del resto del módulo tiene que ser modificado.

La misma interfaz existe para la lista de recursos regulares y la lista de recursos persistentes, tan sólo 'list' se reemplaza por 'plist':

- php3_plist_insert(apuntador, tipo) - devuelve el 'id' del recurso recién insertado
- php3_plist_delete(id) - eliminar el recurso con el id especificado
- php3_plist_find(id,*tipo) - devuelve el apuntador del recurso con el id especificado, actualiza 'tipo' al tipo del recurso

Sin embargo, es más que probable que estas funciones le resulten inútiles cuando intente implementar un módulo persistente. Típicamente, es deseable aprovechar el hecho de que la lista de recursos persistentes es realmente una tabla asociativa. Por ejemplo, en los módulos MySQL/mSQL, cuando hay un llamado a pconnect() (conexión persistente), la función crea una cadena a partir de los valores de host/usuario/contraseña que fueron pasados a la función, y asocia el enlace SQL con ésta cadena como clave. La siguiente vez que alguien haga un llamado a pconnect() con la misma información de host/usuario/contraseña, se generará la misma clave, y la función encontrará el enlace SQL en la lista persistente.

Hasta que sea documentado más a fondo, debería echarle un vistazo a mysql.c o msql.c para ver cómo pueden usarse las capacidades de tabla asociativa de una lista plist.

Una cosa importante a notar: los recursos que van a la lista de recursos persistentes *NO* debe ser reservada con el gestor de memoria de PHP, es decir, NO debe ser creada con emalloc(), estrdup(), etc. En su lugar, deben ser usadas las funciones normales malloc(), strdup(), etc. La razón de esto es simple - al final de la petición (final de cada visita), cada trozo de memoria que fue ubicado usando el gestor de memoria de PHP es eliminado. Ya que la lista persistente no se supone que deba ser eliminada el final de cada petición, no debe utilizarse el gestor de memoria de PHP para reservar recursos que vayan a la lista.

Cuando registra un recurso que va a ser usado en la lista persistente, debe agregar destructores para ésta tanto en la lista no-persistente como en la persistente. El destructor en la lista no-persistente no debería hacer nada. Aquel en la lista persistente debería liberar apropiadamente cualquier recurso obtenido por ese tipo (p.ej. memoria, enlaces SQL, etc). Tal como con los recursos no-persistentes, usted *DEBE* agregar destructores para cada recurso, incluso si no requieren ser destruidos y el destructor puede estar vacío. Recuerde, ya que emalloc() y amigos no deben ser usados junto con la lista persistente, tampoco debe usar efree() aquí.

Añadir directivas de configuración de tiempo de ejecución

Muchas de las características de PHP pueden ser configuradas en tiempo de ejecución. Estas

directivas de configuración pueden aparecer en el archivo php3.ini designado, o, en el caso de la versión módulo de Apache, en los archivos .conf de Apache. La ventaja de tenerlas en los archivos .conf de Apache es que pueden ser configuradas por cada directorio. Esto quiere decir que un directorio puede tener cierto valor para safemodeexecdir, por ejemplo, mientras que otro directorio puede tener otro. Esta especificidad en la configuración es especialmente útil cuando un servidor soporta múltiples hosts virtuales.

Los pasos requeridos para agregar una nueva directiva:

1. Agregar la directiva a la estructura php3_ini_structure en mod_php3.h.

2. En main.c, editar la función php3_module_startup y agregar la llamada apropiada a cfg_get_string() o cfg_get_long().

3. Agregar la directiva, restricciones y un comentario a la estructura php3_commands en mod_php3.c. Fíjese en la parte de restricciones. RSRC_CONF son directivas que pueden estar presentes sólo en las archivos .conf de Apache, Cualquier directiva OR_OPTIONS puede estar presente en cualquier parte, incluyendo archivos `.htaccess` normales.

4. Agregue la entrada apropiada para su directiva en php3take1handler() o en php3flaghandler().

5. En la sección de configuración de la función _php3_info() en functions/info.c necesita agregar su nueva directiva.

6. Y, por último, debe por supuesto usar su directiva en alguna parte. Esta será asequible como php3_ini.directiva.

Llamados a Funciones de Usuario

Para llamar funciones de usuario desde una función interna, debe usar la función **call_user_function()**.

call_user_function() devuelve SUCCESS en caso de éxito, y FAILURE si la función no pudo ser encontrada. ¡Debe chequear ese valor de retorno! Si devuelve SUCCESS, usted es responsable por la destrucción del valor tipo pval retval (o devolverlo como el valor de retorno de su función). Si devuelve FAILURE, el valor de retval será indefinido, y no debe tocarlo.

Todas las funciones internas que hacen llamados a funciones de usuario *deben* ser reentrantes. Entre otras cosas, esto quiere decir que no debe usar variables globales o estáticas.

call_user_function() recibe seis argumentos:

HashTable *tabla_de_funciones

Esta es la tabla asociativa en la que será buscada la función.

---

pval *objeto

Este es un apuntador a un objeto sobre el que es invocada la función. Debe ser **`NULL`** si es llamada una función global. Si no es **`NULL`** (es decir, apunta hacia un objeto), el argumento tabla_de_funciones es ignorado, y en su lugar se toma de la matriz asociativa del objeto. El objeto *puede* ser modificado por la función que es invocada sobre él (la función tendrá acceso sobre el objeto mediante $this). Si por alguna razón no desea que eso ocurra, envíe en su lugar una copia del objeto.

---

pval *nombre_funcion

El nombre de la función a llamar. Debe ser un pval de tipo IS_STRING cuyos miembros function_name.str.val y function_name.str.len se encuentren definidos con valores apropiados. El nombre_funcion es modificado por call_user_function() - es convertido a minúsculas. Si necesita conservar los caracteres originales, envíe una copia del nombre de función en su lugar.

---

pval *retval

Un apuntador a una estructura pval, en la cual es guardado el valor de retorno de la función invocada. La estructura debe ser reservada previamente - **call_user_function()** NO reserva memoria por sí sola.

---

int conteo_param

El número de parámetros que son pasados a la función.

---

pval *params[]

Una matriz de apuntadores a valores que serán pasados como argumentos a la función, el primer argumento ubicado en la posición 0, el segundo en la posición 1, etc. La matriz es una matriz de apuntadores a valores pval; Los apuntadores son enviados tal cual a la función, lo que quiere decir que si la función modifica sus argumentos, los valores originales son modificados (pasados por referencia). Si no desea esta clase de comportamiento, pase una copia en su lugar.

Reporte de Errores

Para reportar errores desde una función interna, debe llamar a la función **php3_error()**. Ésta recibe por lo menos dos parámetros -- el primero es el nivel del error, el segundo es la cadena con formato para el mensaje de error (como en un llamado estándar a **printf()**), y cualquier argumento subsiguiente será usado como parámetro para la cadena de formato. Los niveles de error son:

E_NOTICE

Las noticias no son desplegadas por defecto, e indican que el script encontró algo que puede indicar un error, pero podría ocurrir también en el curso normal de un script en ejecución. Por ejemplo, al tratar de acceder al valor de una variable que no ha sido definida, o al llamar **stat()** sobre un archivo que no existe.

E_WARNING

Las advertencias son desplegadas por defecto, pero no interrumpen la ejecución del script. Éstas indican un problema que debía ser atrapado por el script antes de que el llamado fuera hecho. Por ejemplo, llamar **ereg()** con una expresión regular inválida.

E_ERROR

Los errores son desplegados por defecto también, y la ejecución del script es detenida después de que la función retorna. Éstos indican errores de los que no puede realizarse una recuperación, tales como problemas de reserva de memoria.

E_PARSE

Los errores de análisis sintáctico deben ser generados únicamente por el analizador sintáctico. El código es listado aquí sólo con el propósito de crear una referencia completa.

E_CORE_ERROR

Este es como un **E_ERROR**, excepto que es generado por el núcleo de PHP. Las funciones no deben generar este tipo de error.

E_CORE_WARNING

Este es como un **E_WARNING**, excepto que es generado por el núcleo de PHP. Las funciones no deben generar este tipo de error.

E_COMPILE_ERROR

Este es como un **E_ERROR**, excepto que es generado por el Motor de Scripting de Zend. Las funciones no deben generar este tipo de error.

E_COMPILE_WARNING

Este es como un **E_WARNING**, excepto que es generado por el Motor de Scripting de Zend. Las funciones no deben generar este tipo de error.

E_USER_ERROR

Este es como un **E_ERROR**, excepto que es generado en código PHP usando la función **trigger_error()**. Las funciones no deben generar este tipo de error.

E_USER_WARNING

Este es como un **E_WARNING**, excepto que es generado en código PHP usando la función **trigger_error()**. Las funciones no deben generar este tipo de error.

E_USER_NOTICE

Este es como un **E_NOTICE**, excepto que es generado en código PHP usando la función **trigger_error()**. Las funciones no deben generar este tipo de error.

E_ALL

Todos los anteriores. Usando este nivel de error_reporting le mostrará todos los tipos de error.

Apéndice G. Opciones de configuración

Lista central de opciones de configuración

Abajo está una lista parcial de opciones de configuración usados por los scripts `configure` de PHP cuando se compila en ambientes de tipo UNIX. La mayoría de las opciones de configuración están listadas en su ubicación apropiada en las páginas de referencia de la extensión y no aquí. Para una lista actualizada de las opciones de configuración, ejecute **./configure --help** en su directorio origen de PHP después de ejecutar **autoconf** (vea también el Capítulo de Instalación). También puede interesarle leer la documentación GNU configure para mayor información en otras opciones de **configure** tales como *--prefix=PREFIX*.

> **Nota:** Estas son usadas unicamente en tiempo de compilación. Si desea alterar la configuración en tiempo de ejecución de PHP, por favor vea el capítulo sobre Configuración en tiempo de ejecución.

- Misceláneo
- Comportamiento PHP

- Servidor

Opciones de configuración en PHP 4

Nota: Estas opciones son sólo usadas en PHP 4, tal como PHP 4.1.0. Algunas están disponibles en versiones anteriores a PHP 4, aún algunas en PHP 3, otras solo en PHP 4.1.0. Si quiere compilar una versión anterior, muy probablemente algunas opciones no estarán disponibles.

Opciones Misceláneas

--enable-debug

Compilar con símbolos de rastreo de errores (debugging).

--with-layout=TYPE

Establece como los ficheros instalados serán presentados el tipo uno es de PHP (valor por defecto) o GNU.

--with-pear=DIR

Instala PEAR en DIR (valor por defecto PREFIX/lib/php).

--without-pear

No instale PEAR.

--enable-sigchild

Establece el propio manejador SIGCHLD de PHP.

--disable-rpath

Deshabilita pasar trayectorias de búsqueda de librerias de tiempo de ejecución adicionales.

--enable-libgcc

Habilita explícitamente el ligado con libgcc.

--enable-php-streams

Incluye streams PHP experimentales. ¡No se use a menos que este probando el código!.

--with-zlib-dir[=DIR]

Defina la localización del directorio de instalación de zlib.

--enable-trans-sid

Habilita la propagación transparente del ID de sesión. Válido sólo para PHP 4.1.2 or inferior.

A partir de PHP 4.2.0, la característica trans-id está siempre disponible.

--with-tsrm-pthreads

Usa threads de POSIX (valor por defecto).

--enable-shared[=PKGS]

Construye librerías compartidas [default=yes].

--enable-static[=PKGS]

Construye librerias estáticas [default=yes].

--enable-fast-install[=PKGS]

Optimiza para una rápida instalación [default=yes].

--with-gnu-ld

Asume que el compilador de C usa GNU Id [default=no].

--disable-libtool-lock

Evita bloqueos (debe romper compilaciones paralelas).

--with-pic

Intenta usar sólo objetos PIC/non-PIC [default=use both].

--enable-memory-limit

Compila con soporte de limite de memoria.

--disable-url-fopen-wrapper

Deshabilita empaquetador fopen que abre URLs, que permite accesar ficheros vía HTTP o FTP.

--enable-versioning

Exporta sólo los símbolos requeridos. Vea INSTALL para más información.

--with-imsp[=DIR]

Incluye soporte IMSp (DIR es el directorio de inclusión IMSP's y el directorio de libimsp.a). ¡Sólo PHP3!.

--with-mck[=DIR]

Incluye soporte para Cybercash MCK. DIR es el directorio donde se construye cybercash mck, el directorio por defecto es /usr/src/mck-3.2.0.3-linux, para ayuda sobre esto vea en extra/cyberlib. ¡Sólo PHP 3!.

--with-mod-dav=DIR

Incluye soporte para DAV através del modulo mod_dav de Apache, DIR es el directorio de instalación de mod_dav (sólo el modulo de Apache) ¡Sólo en PHP 3!.

--enable-debugger

Compila con funciones remotas de rastreo de errores. ¡Sólo en PHP 3!.

--enable-versioning

Toma ventaja del alcance y manejo de versiones proveido por Solaris 2.x y Linux. ¡Sólo en PHP 3!.

Opciones de PHP

--enable-maintainer-mode

Habilita las reglas y dependencias de Make, aunque para el que quiera instalar casualmente no son útiles (y a veces hasta pueden confundir).

--with-config-file-path=PATH

Establece la ruta a seguir para `php.ini`, el valor por defecto es /PREFIX/lib.

--enable-safe-mode

Abilita SAFE MODE por defecto.

--with-exec-dir[=DIR]

Solo permite executables en DIR cuando SAFE MODE está por defecto a /usr/local/php/bin.

--enable-magic-quotes

Abilita MAGIC QUOTES por defecto.

--disable-short-tags

Desabilita la forma corta de la etiqueta de inicio php <? por defecto.

---

opciones SAPI

La siguiente lista contiene los SAPI&s (*Server Application Programming Interface*) disponibles para PHP.

--with-aolserver=DIR

Especifica la ruta donde se debe instalar AOLserver.

--with-apxs[=FILE]

Construye un modulo apache compartido. FILE es la ruta opcional a la herramienta apxs de Apache. Asegúrese de específicar la versión de aspx que está instalada en su sistema y NO la que está en el archivo fuente de apache.

--with-apache[=DIR]

Construye un modulo estático de Apache. DIR es el directorio de Apache, por defecto `/usr/local/apache`.

--with-mod_charset

Habilita transferir las tablas para mod_charset (Apache en Ruso).

--with-apxs2[=FILE]

Construye un modulo compartido de Apache 2.0. FILE es la ruta opcional a la herramienta apxs de Apache.

--with-caudium=DIR

Construye PHP como un modulo Pike para ser usado con Caudium. DIR es el directorio del servidor Caudium, con el valor por defecto `/usr/local/caudium/server`.

--disable-cli

Disponible con PHP 4.3.0. Deshabilita contruir la versión CLI de PHP (esto fuerza --without-pear). Más información está disponible en la sección acerca de Usar PHP desde la línea de comandos.

--enable-embed[=TYPE]

Habilita construir la librería embebida SAPI. TYPE es *shared* o *static*, y su valor por defecto es *shared*. Disponible con PHP 4.3.0.

--with-fhttpd[=DIR]

Cosntruye el modulo fhttpd. DIR es el directorio de los fuentes de fhttpd, su valor por defecto es `/usr/local/src/fhttpd`. Ya no está disponible a partir de PHP 4.3.0.

--with-isapi=DIR

Construye PHP como un modulo ISAPI para ser usado con Zeus.

--with-nsapi=DIR

Específica la ruta al servidor web instalado Netscape/iPlanet/SunONE.

--with-phttpd=DIR

Sin información.

--with-pi3web=DIR

Cosntruye PHP como un modulo para ser usado con Pi3Web.

--with-roxen=DIR

Cosntruye PHP como un modulo Pike. DIR es el directorio base de Roxen, normalmente `/usr/local/roxen/server`.

--enable-roxen-zts

Construye el modulo Roxen usando "Zend Thread Safety".

--with-servlet[=DIR]

Incluye soporte para servlet. DIR es el directorio de instalación base para JSDK. Este SAPI requiere que la extensión de java sea construida como un dl compartido.

--with-thttpd=SRCDIR

Construye PHP como un modulo thttpd.

--with-tux=MODULEDIR

Construye PHP como un modulo TUX (sólo Linux).

--with-webjames=SRCDIR

Construye PHP como un modulo WebJames (sólo RISC OS).

--disable-cgi

Deshabilita la construcción de la versión CGI de PHP. Disponible con PHP 4.3.0.

--enable-force-cgi-redirect

Habilita el chequeo de seguridad para redirecciones internas del servidor. Debe usar este si está ejecutando la versión CGI con Apache.

--enable-discard-path

Si está habilitado, el binario CGI de PHP puede se puesto fuera del arbol web en forma segura y la gente no será capaz de evitar la seguridad de `.htaccess`

--with-fastcgi

Cosntruye PHP como una aplicación FastCGI. Ya no está disponible desde PHP 4.3.0, en lugar de esto usted debe usar *--enable-fastcgi*.

--enable-fastcgi

Si está habilitado, el modulo CGI será construido con soporte para FastCGI también. Disponible desde PHP 4.3.0.

--disable-path-info-check

Si no está habilitado, fallarán las rutas tales como `/info.php/test?a=b`. Disponible desde PHP 4.3.0. Para mÃ‚Â´s información vea el Manual de Apache.

Apéndice H. Directivas de `php.ini`

Lista de directivas de `php.ini`

Esta lista incluye las directivas de `php.ini` que puede definir para configurar su instalación de PHP.

Tabla H-1. Opciones de configuración

Nombre	Predeterminado	Modificable	Cambios
allow_call_time_pass_reference	"1"	PHP_INI_PERDIR	PHP_INI_ALL en PHP <= 4.0.0.
allow_url_fopen	"1"	PHP_INI_SYSTEM	PHP_INI_ALL en PHP <= 4.3.4. Disponible a partir de PHP 4.0.4.
always_populate_raw_post_data	"0"	PHP_INI_PERDIR	PHP_INI_ALL en PHP <= 4.2.3. Disponible a partir de PHP 4.1.0.
apc.cache_by_default	"1"	PHP_INI_SYSTEM	
apc.enabled	"1"	PHP_INI_SYSTEM	
apc.filters	""	PHP_INI_SYSTEM	
apc.gc_ttl	"3600"	PHP_INI_SYSTEM	
apc.mmap_file_mask	NULL	PHP_INI_SYSTEM	
apc.num_files_hint	"1000"	PHP_INI_SYSTEM	
apc.optimization	"0"	PHP_INI_SYSTEM	
apc.shm_segments	"1"	PHP_INI_SYSTEM	
apc.shm_size	"30"	PHP_INI_SYSTEM	
apc.slam_defense	"0"	PHP_INI_SYSTEM	
apc.ttl	"0"	PHP_INI_SYSTEM	
apc.user_entries_hint	"100"	PHP_INI_SYSTEM	
apc.user_ttl	"0"	PHP_INI_SYSTEM	

Nombre	Predeterminado	Modificable	Cambios
apd.dumpdir	NULL	PHP_INI_ALL	
apd.statement_tracing	"0"	PHP_INI_ALL	
arg_separator.input	"&"	PHP_INI_PERDIR	Disponible a partir de PHP 4.0.5.
arg_separator.output	"&"	PHP_INI_ALL	Disponible a partir de PHP 4.0.5.
asp_tags	"0"	PHP_INI_PERDIR	PHP_INI_ALL en PHP <= 4.0.0.
assert.active	"1"	PHP_INI_ALL	
assert.bail	"0"	PHP_INI_ALL	
assert.callback	NULL	PHP_INI_ALL	
assert.quiet_eval	"0"	PHP_INI_ALL	
assert.warning	"1"	PHP_INI_ALL	
auto_append_file	NULL	PHP_INI_PERDIR	PHP_INI_ALL en PHP <= 4.2.3.
auto_detect_line_endings	"0"	PHP_INI_ALL	Disponible a partir de PHP 4.3.0.
auto_globals_jit	"1"	PHP_INI_PERDIR	Disponible a partir de PHP 5.0.0.
auto_prepend_file	NULL	PHP_INI_PERDIR	PHP_INI_ALL en PHP <= 4.2.3.
bcmath.scale	"0"	PHP_INI_ALL	Disponible a partir de PHP 5.0.0.
blenc.key_file	"/usr/local/etc/blenckeys"	PHP_INI_ALL	
browscap	NULL	PHP_INI_SYSTEM	
child_terminate	"0"	PHP_INI_ALL	Disponible a partir de PHP 4.0.5.
com.allow_dcom	"0"	PHP_INI_SYSTEM	Disponible a partir de PHP 4.0.5.
com.autoregister_casesensitive	"1"	PHP_INI_ALL	PHP_INI_SYSTEM en PHP 4. Disponible a partir de PHP 4.1.0.
com.autoregister_typelib	"0"	PHP_INI_ALL	PHP_INI_SYSTEM en PHP 4. Disponible a partir de PHP 4.1.0.

Nombre	Predeterminado	Modificable	Cambios
com.autoregister_verbose	"0"	PHP_INI_ALL	PHP_INI_SYSTEM en PHP 4. Disponible a partir de PHP 4.1.0.
com.code_page	""	PHP_INI_ALL	Disponible a partir de PHP 5.0.0.
com.typelib_file	""	PHP_INI_SYSTEM	Disponible a partir de PHP 4.0.5.
crack.default_dictionary	NULL	PHP_INI_SYSTEM	Disponible a partir de PHP 4.0.5.
daffodildb.default_host	"localhost"	PHP_INI_ALL	
daffodildb.default_password	"daffodil"	PHP_INI_ALL	
daffodildb.default_socket	NULL	PHP_INI_ALL	
daffodildb.default_user	"DAFFODIL"	PHP_INI_ALL	
daffodildb.port	"3456"	PHP_INI_ALL	
date.default_latitude	"31.7667"	PHP_INI_ALL	Disponible a partir de PHP 5.0.0.
date.default_longitude	"35.2333"	PHP_INI_ALL	Disponible a partir de PHP 5.0.0.
date.sunrise_zenith	"90.83"	PHP_INI_ALL	Disponible a partir de PHP 5.0.0.
date.sunset_zenith	"90.83"	PHP_INI_ALL	Disponible a partir de PHP 5.0.0.
dba.default_handler	""	PHP_INI_ALL	Disponible a partir de PHP 4.3.3.
dbx.colnames_case	"unchanged"	PHP_INI_SYSTEM	Disponible a partir de PHP 4.3.0.
default_charset	""	PHP_INI_ALL	
default_mimetype	"text/html"	PHP_INI_ALL	
default_socket_timeout	"60"	PHP_INI_ALL	Disponible a partir de PHP 4.3.0.
define_syslog_variables	"0"	PHP_INI_ALL	
disable_classes	""	Solo en `php.ini`	Disponible a partir de PHP 4.3.2.
disable_functions	""	Solo en `php.ini`	Disponible a partir de PHP 4.0.1.

Nombre	Predeterminado	Modificable	Cambios
display_errors	"1"	PHP_INI_ALL	
display_startup_errors	"0"	PHP_INI_ALL	Disponible a partir de PHP 4.0.3.
docref_ext	""	PHP_INI_ALL	Disponible a partir de PHP 4.3.2.
docref_root	""	PHP_INI_ALL	Disponible a partir de PHP 4.3.0.
doc_root	NULL	PHP_INI_SYSTEM	
enable_dl	"1"	PHP_INI_SYSTEM	
engine	"1"	PHP_INI_ALL	Disponible a partir de PHP 4.0.5.
error_append_string	NULL	PHP_INI_ALL	
error_log	NULL	PHP_INI_ALL	
error_prepend_string	NULL	PHP_INI_ALL	
error_reporting	NULL	PHP_INI_ALL	
exif.decode_jis_intel	"JIS"	PHP_INI_ALL	Disponible a partir de PHP 4.3.0.
exif.decode_jis_motorola	"JIS"	PHP_INI_ALL	Disponible a partir de PHP 4.3.0.
exif.decode_unicode_intel	"UCS-2LE"	PHP_INI_ALL	Disponible a partir de PHP 4.3.0.
exif.decode_unicode_motorola	"UCS-2BE"	PHP_INI_ALL	Disponible a partir de PHP 4.3.0.
exif.encode_jis	""	PHP_INI_ALL	Disponible a partir de PHP 4.3.0.
exif.encode_unicode	"ISO-8859-15"	PHP_INI_ALL	Disponible a partir de PHP 4.3.0.
expose_php	"1"	Solo en `php.ini`	
extension_dir	"/path/to/php"	PHP_INI_SYSTEM	
fbsql.allow_persistent	"1"	PHP_INI_SYSTEM	Disponible a partir de PHP 4.2.0.
fbsql.autocommit	"1"	PHP_INI_SYSTEM	Disponible a partir de PHP 4.0.6.
fbsql.batchsize	"1000"	PHP_INI_ALL	Disponible a partir de PHP 5-cvs.

Nombre	Predeterminado	Modificable	Cambios
fbsql.default_database	""	PHP_INI_SYSTEM	Disponible a partir de PHP 4.0.6.
fbsql.default_database_password	""	PHP_INI_SYSTEM	Disponible a partir de PHP 4.0.6.
fbsql.default_host	NULL	PHP_INI_SYSTEM	Disponible a partir de PHP 4.0.6.
fbsql.default_password	""	PHP_INI_SYSTEM	Disponible a partir de PHP 4.0.6.
fbsql.default_user	"_SYSTEM"	PHP_INI_SYSTEM	Disponible a partir de PHP 4.0.6.
fbsql.generate_warnings	"0"	PHP_INI_SYSTEM	Disponible a partir de PHP 4.0.6.
fbsql.max_connections	"128"	PHP_INI_SYSTEM	Disponible a partir de PHP 4.0.6.
fbsql.max_links	"128"	PHP_INI_SYSTEM	Disponible a partir de PHP 4.0.6.
fbsql.max_persistent	"-1"	PHP_INI_SYSTEM	Disponible a partir de PHP 4.0.6.
fbsql.max_results	"128"	PHP_INI_SYSTEM	Disponible a partir de PHP 4.0.6.
file_uploads	"1"	PHP_INI_SYSTEM	PHP_INI_ALL en PHP <= 4.2.3. Disponible a partir de PHP 4.0.3.
highlight.bg	"#FFFFFF"	PHP_INI_ALL	
highlight.comment	"#FF8000"	PHP_INI_ALL	
highlight.default	"#0000BB"	PHP_INI_ALL	
highlight.html	"#000000"	PHP_INI_ALL	
highlight.keyword	"#007700"	PHP_INI_ALL	
highlight.string	"#DD0000"	PHP_INI_ALL	
html_errors	"1"	PHP_INI_ALL	PHP_INI_SYSTEM en PHP <= 4.2.3. Disponible a partir de PHP 4.0.2.
hyperwave.allow_persistent	"0"	PHP_INI_SYSTEM	Disponible a partir de PHP 4.3.2.
hyperwave.default_port	"418"	PHP_INI_ALL	
ibase.allow_persistent	"1"	PHP_INI_SYSTEM	

Nombre	Predeterminado	Modificable	Cambios
ibase.dateformat	"%Y-%m-%d"	PHP_INI_ALL	
ibase.default_charset	NULL	PHP_INI_ALL	Disponible a partir de PHP 5.0.0.
ibase.default_db	NULL	PHP_INI_SYSTEM	Disponible a partir de PHP 5.0.0.
ibase.default_password	NULL	PHP_INI_ALL	
ibase.default_user	NULL	PHP_INI_ALL	
ibase.max_links	"-1"	PHP_INI_SYSTEM	
ibase.max_persistent	"-1"	PHP_INI_SYSTEM	
ibase.timeformat	"%H:%M:%S"	PHP_INI_ALL	
ibase.timestampformat	"%Y-%m-%d %H:%M:%S"	PHP_INI_ALL	
iconv.input_encoding	"ISO-8859-1"	PHP_INI_ALL	Disponible a partir de PHP 4.0.5.
iconv.internal_encoding	"ISO-8859-1"	PHP_INI_ALL	Disponible a partir de PHP 4.0.5.
iconv.output_encoding	"ISO-8859-1"	PHP_INI_ALL	Disponible a partir de PHP 4.0.5.
ifx.allow_persistent	"1"	PHP_INI_SYSTEM	
ifx.blobinfile	"1"	PHP_INI_ALL	
ifx.byteasvarchar	"0"	PHP_INI_ALL	
ifx.charasvarchar	"0"	PHP_INI_ALL	
ifx.default_host	NULL	PHP_INI_SYSTEM	
ifx.default_password	NULL	PHP_INI_SYSTEM	
ifx.default_user	NULL	PHP_INI_SYSTEM	
ifx.max_links	"-1"	PHP_INI_SYSTEM	
ifx.max_persistent	"-1"	PHP_INI_SYSTEM	
ifx.nullformat	"0"	PHP_INI_ALL	

Nombre	Predeterminado	Modificable	Cambios
ifx.textasvarchar	"0"	PHP_INI_ALL	
ignore_repeated_errors	"0"	PHP_INI_ALL	Disponible a partir de PHP 4.3.0.
ignore_repeated_source	"0"	PHP_INI_ALL	Disponible a partir de PHP 4.3.0.
ignore_user_abort	"0"	PHP_INI_ALL	
implicit_flush	"0"	PHP_INI_ALL	PHP_INI_PERDIR en PHP <= 4.2.3.
include_path	".;/path/to/php/pear"	PHP_INI_ALL	
ingres.allow_persistent	"1"	PHP_INI_SYSTEM	Disponible a partir de PHP 4.0.2.
ingres.default_database	NULL	PHP_INI_ALL	Disponible a partir de PHP 4.0.2.
ingres.default_password	NULL	PHP_INI_ALL	Disponible a partir de PHP 4.0.2.
ingres.default_user	NULL	PHP_INI_ALL	Disponible a partir de PHP 4.0.2.
ingres.max_links	"-1"	PHP_INI_SYSTEM	Disponible a partir de PHP 4.0.2.
ingres.max_persistent	"-1"	PHP_INI_SYSTEM	Disponible a partir de PHP 4.0.2.
ircg.control_user	"nobody"	PHP_INI_ALL	Disponible a partir de PHP 5.0.0.
ircg.keep_alive_interval	"60"	PHP_INI_ALL	Disponible a partir de PHP 5.0.0.
ircg.max_format_message_sets	"12"	PHP_INI_ALL	Disponible a partir de PHP 5.0.0.
ircg.shared_mem_size	"6000000"	PHP_INI_ALL	Disponible a partir de PHP 5.0.0.
ircg.work_dir	"/tmp/ircg"	PHP_INI_ALL	Disponible a partir de PHP 5.0.0.
last_modified	"0"	PHP_INI_ALL	Disponible a partir de PHP 4.0.5.
ldap.max_links	"-1"	PHP_INI_SYSTEM	
log_errors	"0"	PHP_INI_ALL	
log_errors_max_len	"1024"	PHP_INI_ALL	Disponible a partir de PHP 4.3.0.
magic_quotes_gpc	"1"	PHP_INI_PERDIR	PHP_INI_ALL en PHP <= 4.2.3.

Nombre	Predeterminado	Modificable	Cambios
magic_quotes_runtime	"0"	PHP_INI_ALL	
magic_quotes_sybase	"0"	PHP_INI_ALL	
mail.force_extra_parameters	NULL	PHP_INI_PERDIR	Disponible a partir de PHP 5.0.0.
mailparse.def_charset	"us-ascii"	PHP_INI_ALL	Disponible a partir de PHP 4.1.0.
maxdb.default_db	NULL	PHP_INI_ALL	
maxdb.default_host	NULL	PHP_INI_ALL	
maxdb.default_pw	NULL	PHP_INI_ALL	
maxdb.default_user	NULL	PHP_INI_ALL	
maxdb.long_readlen	"200"	PHP_INI_ALL	
max_execution_time	"30"	PHP_INI_ALL	
max_input_time	"-1"	PHP_INI_PERDIR	Disponible a partir de PHP 4.3.0.
mbstring.detect_order	NULL	PHP_INI_ALL	Disponible a partir de PHP 4.0.6.
mbstring.encoding_translation	"0"	PHP_INI_PERDIR	Disponible a partir de PHP 4.3.0.
mbstring.func_overload	"0"	PHP_INI_PERDIR	PHP_INI_SYSTEM en PHP <= 4.2.3. Disponible a partir de PHP 4.2.0.
mbstring.http_input	"pass"	PHP_INI_ALL	Disponible a partir de PHP 4.0.6.
mbstring.http_output	"pass"	PHP_INI_ALL	Disponible a partir de PHP 4.0.6.
mbstring.internal_encoding	NULL	PHP_INI_ALL	Disponible a partir de PHP 4.0.6.
mbstring.language	"neutral"	PHP_INI_PERDIR	Disponible a partir de PHP 4.3.0.
mbstring.script_encoding	NULL	PHP_INI_ALL	Disponible a partir de PHP 4.3.0.
mbstring.substitute_character	NULL	PHP_INI_ALL	Disponible a partir de PHP 4.0.6.
mcrypt.algorithms_dir	NULL	PHP_INI_ALL	Disponible a partir de PHP 4.0.2.

Nombre	Predeterminado	Modificable	Cambios
mcrypt.modes_dir	NULL	PHP_INI_ALL	Disponible a partir de PHP 4.0.2.
memory_limit	"8M"	PHP_INI_ALL	
mime_magic.debug	"0"	PHP_INI_SYSTEM	Disponible a partir de PHP 5.0.0.
mime_magic.magicfile	"/path/to/php/magic.mime"	PHP_INI_SYSTEM	Disponible a partir de PHP 4.3.0.
mssql.allow_persistent	"1"	PHP_INI_SYSTEM	
mssql.batchsize	"0"	PHP_INI_ALL	Disponible a partir de PHP 4.0.4.
mssql.compatability_mode	"0"	PHP_INI_ALL	
mssql.connect_timeout	"5"	PHP_INI_ALL	
mssql.datetimeconvert	"1"	PHP_INI_ALL	Disponible a partir de PHP 4.2.0.
mssql.max_links	"-1"	PHP_INI_SYSTEM	
mssql.max_persistent	"-1"	PHP_INI_SYSTEM	
mssql.max_procs	"25"	PHP_INI_ALL	Disponible a partir de PHP 4.3.0.
mssql.min_error_severity	"10"	PHP_INI_ALL	
mssql.min_message_severity	"10"	PHP_INI_ALL	
mssql.secure_connection	"0"	PHP_INI_SYSTEM	Disponible a partir de PHP 4.3.0.
mssql.textlimit	"-1"	PHP_INI_ALL	
mssql.textsize	"-1"	PHP_INI_ALL	
mssql.timeout	"60"	PHP_INI_ALL	Disponible a partir de PHP 4.1.0.
mysql.allow_persistent	"1"	PHP_INI_SYSTEM	
mysql.connect_timeout	"60"	PHP_INI_ALL	PHP_INI_SYSTEM en PHP <= 4.3.2. Disponible a partir de PHP 4.3.0.
mysql.default_host	NULL	PHP_INI_ALL	

Nombre	Predeterminado	Modificable	Cambios
mysql.default_password	NULL	PHP_INI_ALL	
mysql.default_port	NULL	PHP_INI_ALL	
mysql.default_socket	NULL	PHP_INI_ALL	Disponible a partir de PHP 4.0.1.
mysql.default_user	NULL	PHP_INI_ALL	
mysql.max_links	"-1"	PHP_INI_SYSTEM	
mysql.max_persistent	"-1"	PHP_INI_SYSTEM	
mysql.trace_mode	"0"	PHP_INI_ALL	Disponible a partir de PHP 4.3.0.
mysqli.default_host	NULL	PHP_INI_ALL	Disponible a partir de PHP 5.0.0.
mysqli.default_port	"3306"	PHP_INI_ALL	Disponible a partir de PHP 5.0.0.
mysqli.default_pw	NULL	PHP_INI_ALL	Disponible a partir de PHP 5.0.0.
mysqli.default_socket	NULL	PHP_INI_ALL	Disponible a partir de PHP 5.0.0.
mysqli.default_user	NULL	PHP_INI_ALL	Disponible a partir de PHP 5.0.0.
mysqli.max_links	"-1"	PHP_INI_SYSTEM	Disponible a partir de PHP 5.0.0.
mysqli.reconnect	"0"	PHP_INI_SYSTEM	Disponible a partir de PHP 5.0.0.
namazu.debugmode	"0"	PHP_INI_ALL	
namazu.lang	NULL	PHP_INI_ALL	
namazu.loggingmode	"0"	PHP_INI_ALL	
namazu.sortmethod	NULL	PHP_INI_ALL	
namazu.sortorder	NULL	PHP_INI_ALL	
nsapi.read_timeout	"60"	PHP_INI_ALL	Disponible a partir de PHP 4.3.3.
odbc.allow_persistent	"1"	PHP_INI_SYSTEM	
odbc.check_persistent	"1"	PHP_INI_SYSTEM	

Nombre	Predeterminado	Modificable	Cambios
odbc.defaultbinmode	"1"	PHP_INI_ALL	
odbc.defaultlrl	"4096"	PHP_INI_ALL	
odbc.default_db	NULL	PHP_INI_ALL	
odbc.default_pw	NULL	PHP_INI_ALL	
odbc.default_user	NULL	PHP_INI_ALL	
odbc.max_links	"-1"	PHP_INI_SYSTEM	
odbc.max_persistent	"-1"	PHP_INI_SYSTEM	
opendirectory.max_refs	"-1"	PHP_INI_ALL	
opendirectory.separator	"/"	PHP_INI_ALL	
open_basedir	NULL	PHP_INI_SYSTEM	
output_buffering	"0"	PHP_INI_PERDIR	
output_handler	NULL	PHP_INI_PERDIR	Disponible a partir de PHP 4.0.4.
pdo.global_value	"42"	PHP_INI_ALL	
pfpro.defaulthost	"test-payflow.verisign.com"	PHP_INI_ALL	Disponible a partir de PHP 4.0.2.
pfpro.defaultport	"443"	PHP_INI_ALL	Disponible a partir de PHP 4.0.2.
pfpro.defaulttimeout	"30"	PHP_INI_ALL	Disponible a partir de PHP 4.0.2.
pfpro.proxyaddress	""	PHP_INI_ALL	Disponible a partir de PHP 4.0.2.
pfpro.proxylogon	""	PHP_INI_ALL	Disponible a partir de PHP 4.0.2.
pfpro.proxypassword	""	PHP_INI_ALL	Disponible a partir de PHP 4.0.2.
pfpro.proxyport	""	PHP_INI_ALL	Disponible a partir de PHP 4.0.2.
pgsql.allow_persistent	"1"	PHP_INI_SYSTEM	
pgsql.auto_reset_persistent	"0"	PHP_INI_SYSTEM	Disponible a partir de PHP 4.2.0.

Nombre	Predeterminado	Modificable	Cambios
pgsql.ignore_notice	"0"	PHP_INI_ALL	Disponible a partir de PHP 4.3.0.
pgsql.log_notice	"0"	PHP_INI_ALL	Disponible a partir de PHP 4.3.0.
pgsql.max_links	"-1"	PHP_INI_SYSTEM	
pgsql.max_persistent	"-1"	PHP_INI_SYSTEM	
post_max_size	"8M"	PHP_INI_PERDIR	PHP_INI_SYSTEM en PHP <= 4.2.3. Disponible a partir de PHP 4.0.3.
precision	"14"	PHP_INI_ALL	
printer.default_printer	""	PHP_INI_ALL	
realpath_cache_size	"16K"	PHP_INI_SYSTEM	Disponible a partir de PHP 5-cvs.
realpath_cache_ttl	"120"	PHP_INI_SYSTEM	Disponible a partir de PHP 5-cvs.
register_argc_argv	"1"	PHP_INI_PERDIR	PHP_INI_ALL en PHP <= 4.2.3.
register_globals	"0"	PHP_INI_PERDIR	PHP_INI_ALL en PHP <= 4.2.3.
register_long_arrays	"1"	PHP_INI_PERDIR	Disponible a partir de PHP 5.0.0.
report_memleaks	"1"	PHP_INI_ALL	Disponible a partir de PHP 4.3.0.
report_zend_debug	"1"	PHP_INI_ALL	Disponible a partir de PHP 5.0.0.
safe_mode	"0"	PHP_INI_SYSTEM	
safe_mode_allowed_env_vars	"PHP_"	PHP_INI_SYSTEM	
safe_mode_exec_dir	""	PHP_INI_SYSTEM	
safe_mode_gid	"0"	PHP_INI_SYSTEM	Disponible a partir de PHP 4.1.0.
safe_mode_include_dir	NULL	PHP_INI_SYSTEM	Disponible a partir de PHP 4.1.0.
safe_mode_protected_env_vars	"LD_LIBRARY_PATH"	PHP_INI_SYSTEM	
sendmail_from	NULL	PHP_INI_ALL	

Nombre	Predeterminado	Modificable	Cambios
sendmail_path	NULL	PHP_INI_SYSTEM	
serialize_precision	"100"	PHP_INI_ALL	Disponible a partir de PHP 4.3.2.
session.auto_start	"0"	PHP_INI_ALL	
session.bug_compat_42	"1"	PHP_INI_ALL	Disponible a partir de PHP 4.3.0.
session.bug_compat_warn	"1"	PHP_INI_ALL	Disponible a partir de PHP 4.3.0.
session.cache_expire	"180"	PHP_INI_ALL	
session.cache_limiter	"nocache"	PHP_INI_ALL	
session.cookie_domain	""	PHP_INI_ALL	
session.cookie_lifetime	"0"	PHP_INI_ALL	
session.cookie_path	"/"	PHP_INI_ALL	
session.cookie_secure	""	PHP_INI_ALL	Disponible a partir de PHP 4.0.4.
session.entropy_file	""	PHP_INI_ALL	
session.entropy_length	"0"	PHP_INI_ALL	
session.gc_divisor	"100"	PHP_INI_ALL	Disponible a partir de PHP 4.3.2.
session.gc_maxlifetime	"1440"	PHP_INI_ALL	
session.gc_probability	"1"	PHP_INI_ALL	
session.hash_bits_per_character	"4"	PHP_INI_ALL	Disponible a partir de PHP 5.0.0.
session.hash_function	"0"	PHP_INI_ALL	Disponible a partir de PHP 5.0.0.
session.name	"PHPSESSID"	PHP_INI_ALL	
session.referer_check	""	PHP_INI_ALL	
session.save_handler	"files"	PHP_INI_ALL	
session.save_path	""	PHP_INI_ALL	

Nombre	Predeterminado	Modificable	Cambios
session.serialize_handler	"php"	PHP_INI_ALL	
session.use_cookies	"1"	PHP_INI_ALL	
session.use_only_cookies	"0"	PHP_INI_ALL	Disponible a partir de PHP 4.3.0.
session.use_trans_sid	"0"	PHP_INI_ALL	PHP_INI_ALL en PHP <= 4.2.3. PHP_INI_PERDIR en PHP <= 4-cvs. Disponible a partir de PHP 4.0.3.
session_pgsql.create_table	"1"	PHP_INI_SYSTEM	
session_pgsql.db	"host=localhost dbname=php_session user=nobody"	PHP_INI_SYSTEM	
session_pgsql.disable	"0"	PHP_INI_SYSTEM	
session_pgsql.failover_mode	"0"	PHP_INI_SYSTEM	
session_pgsql.gc_interval	"3600"	PHP_INI_SYSTEM	
session_pgsql.keep_expired	"0"	PHP_INI_SYSTEM	
session_pgsql.sem_file_name	"/tmp/php_session_pgsql"	PHP_INI_SYSTEM	
session_pgsql.serializable	"0"	PHP_INI_SYSTEM	
session_pgsql.short_circuit	"0"	PHP_INI_SYSTEM	
session_pgsql.use_app_vars	"0"	PHP_INI_SYSTEM	
session_pgsql.vacuum_interval	"21600"	PHP_INI_SYSTEM	
short_open_tag	"1"	PHP_INI_PERDIR	PHP_INI_ALL en PHP <= 4.0.0.
simple_cvs.authMethod	"0"	PHP_INI_ALL	
simple_cvs.compressionLevel	"0"	PHP_INI_ALL	
simple_cvs.cvsRoot	"0"	PHP_INI_ALL	
simple_cvs.host	"0"	PHP_INI_ALL	

Nombre	Predeterminado	Modificable	Cambios
simple_cvs.moduleName	"0"	PHP_INI_ALL	
simple_cvs.userName	"0"	PHP_INI_ALL	
simple_cvs.workingDir	"0"	PHP_INI_ALL	
SMTP	"localhost"	PHP_INI_ALL	
smtp_port	"25"	PHP_INI_ALL	Disponible a partir de PHP 4.3.0.
soap.wsdl_cache_dir	"/tmp"	PHP_INI_ALL	Disponible a partir de PHP 5.0.0.
soap.wsdl_cache_enabled	"1"	PHP_INI_ALL	Disponible a partir de PHP 5.0.0.
soap.wsdl_cache_ttl	"86400"	PHP_INI_ALL	Disponible a partir de PHP 5.0.0.
sql.safe_mode	"0"	PHP_INI_SYSTEM	
sqlite.assoc_case	"0"	PHP_INI_ALL	Disponible a partir de PHP 5.0.0.
sybct.allow_persistent	"1"	PHP_INI_SYSTEM	Disponible a partir de PHP 4.0.4.
sybct.deadlock_retry_count	"0"	PHP_INI_ALL	Disponible a partir de PHP 4.3.0.
sybct.hostname	NULL	PHP_INI_ALL	Disponible a partir de PHP 4.0.4.
sybct.login_timeout	"-1"	PHP_INI_ALL	Disponible a partir de PHP 4.3.5.
sybct.max_links	"-1"	PHP_INI_SYSTEM	Disponible a partir de PHP 4.0.4.
sybct.max_persistent	"-1"	PHP_INI_SYSTEM	Disponible a partir de PHP 4.0.4.
sybct.min_client_severity	"10"	PHP_INI_ALL	Disponible a partir de PHP 4.0.4.
sybct.min_server_severity	"10"	PHP_INI_ALL	Disponible a partir de PHP 4.0.4.
tidy.clean_output	"0"	PHP_INI_PERDIR	Disponible a partir de PHP 5.0.0.
tidy.default_config	""	PHP_INI_SYSTEM	Disponible a partir de PHP 5.0.0.
track_errors	"0"	PHP_INI_ALL	
unserialize_callback_func	NULL	PHP_INI_ALL	Disponible a partir de PHP 4.2.0.

Nombre	Predeterminado	Modificable	Cambios
upload_max_filesize	"2M"	PHP_INI_PERDIR	PHP_INI_ALL en PHP <= 4.2.3.
upload_tmp_dir	NULL	PHP_INI_SYSTEM	
url_rewriter.tags	"a=href,area=href,frame=src,form=,fieldset="	PHP_INI_ALL	Disponible a partir de PHP 4.0.4.
user_agent	NULL	PHP_INI_ALL	Disponible a partir de PHP 4.3.0.
user_dir	NULL	PHP_INI_SYSTEM	
valkyrie.auto_validate	"0"	PHP_INI_ALL	
valkyrie.config_path	NULL	PHP_INI_ALL	
variables_order	"EGPCS"	PHP_INI_ALL	
xbithack	"0"	PHP_INI_ALL	Disponible a partir de PHP 4.0.5.
xmlrpc_errors	"0"	PHP_INI_SYSTEM	Disponible a partir de PHP 4.1.0.
xmlrpc_error_number	"0"	PHP_INI_ALL	Disponible a partir de PHP 4.1.0.
xmms.path	"/usr/bin/xmms"	PHP_INI_ALL	
xmms.session	"0"	PHP_INI_ALL	
y2k_compliance	"1"	PHP_INI_ALL	
yaz.keepalive	"120"	PHP_INI_ALL	
yaz.log_file	NULL	PHP_INI_ALL	Disponible a partir de PHP 4.3.0.
yaz.max_links	"100"	PHP_INI_ALL	Disponible a partir de PHP 4.3.0.
zend.ze1_compatibility_mode	"0"	PHP_INI_ALL	Disponible a partir de PHP 5.0.0.
zlib.output_compression	"0"	PHP_INI_ALL	Disponible a partir de PHP 4.0.5.
zlib.output_compression_level	"-1"	PHP_INI_ALL	Disponible a partir de PHP 4.3.0.
zlib.output_handler	""	PHP_INI_ALL	Disponible a partir de PHP 4.3.0.

Tabla H-2. Definición de constantes PHP_INI_*

Constante	Valor	Significado
PHP_INI_USER	1	La entrada puede definirse en scripts de usuario
PHP_INI_PERDIR	2	La entrada puede definirse en `php.ini`, `.htaccess` o `httpd.conf`
PHP_INI_SYSTEM	4	La entrada puede definirse en `php.ini` o `httpd.conf`
PHP_INI_ALL	7	La entrada puede definirse en cualquier parte

Descripción de las directivas de núcleo en `php.ini`

Esta lista incluye las directivas de núcleo en `php.ini` que puede definir para configurar su instalación de PHP. Las directivas gestionadas por extensiones son listadas en las páginas de documentación de las extensiones respectivamente. Por ejemplo, puede encontrarse información sobre las directivas de sesiones en la página de sesiones.

Opciones Httpd

Tabla H-3. Opciones Httpd

Nombre	Predeterminado	Modificable
async_send	"0"	PHP_INI_ALL

Opciones del Lenguaje

Tabla H-4. Opciones del Lenguaje y Configuración Variada

Nombre	Predeterminado	Modificable
short_open_tag	On	PHP_INI_SYSTEM| PHP_INI_PERDIR
asp_tags	Off	PHP_INI_SYSTEM| PHP_INI_PERDIR
precision	"14"	PHP_INI_ALL
y2k_compliance	Off	PHP_INI_ALL
allow_call_time_pass_reference	On	PHP_INI_SYSTEM| PHP_INI_PERDIR
expose_php	On	PHP_INI_SYSTEM
zend.ze1_compatibility_mode	Off	PHP_INI_ALL

A continuación se presenta una corta explicación de las directivas de configuración.

short_open_tag **boolean**

Indica si se permite el uso de la forma corta (`<? ?>`) de la etiqueta de apertura de PHP. Si desea usar PHP en conjunto con XML, puede deshabilitar esta opción, de modo que pueda

usar `<?xml ?>` en forma directa. De otro modo, puede imprimir esta cadena con PHP, por ejemplo: `<?php echo '<?xml version="1.0"'; ?>`. Asimismo, si el parámetro está deshabilitado, debe usar la forma larga de la etiqueta de apertura de PHP (`<?php ?>`).

> **Nota:** Esta directiva afecta también la contracción `<?=`, la cual es idéntica a `<? echo`. El uso de este atajo requiere que short_open_tag se encuentre habilitado.

asp_tags **boolean**

Habilita el uso de etiquetas <% %> tipo-ASP además de las etiquetas convencionales <?php ?>. Esto incluye el atajo para imprimir valores de variable <%= $valor %>. Para más información, vea Escapar desde HTML.

> **Nota:** El soporte para etiquetas tipo-ASP fue agregado en 3.0.4.

precision **integer**

El número de dígitos significativos desplegados en números de punto flotante.

y2k_compliance **boolean**

Obligar compatibilidad con el año 2000 (causa problemas con navegadores no-compatibles)

allow_call_time_pass_reference **boolean**

Indica si se permite la habilidad de obligar que los argumentos sean pasados por referencia al momento de efectuar llamados de función. Este método es considerado obsoleto y es posible que no sea soportado en versiones futuras de PHP/Zend. El método recomendado de especificar cuáles argumentos deben ser pasados por referencia se encuentra en la declaración de funciones. Es recomendable que proceda a deshabilitar esta opción y asegurarse de que sus scripts trabajen correctamente sin ella, de modo que pueda estar seguro de que trabajarán con versiones futuras del lenguaje (recibirá una advertencia cada vez que use esta característica, y el argumento será pasado por valor, y no por referencia).

Pasar argumentos por referencia al momento de llamar una función fue declarado obsoleto por razones de limpieza del código. La función puede modificar su argumento de una forma no documentada si no se declara que el argumento es pasado por referencia. Para prevenir efectos colaterales, es mejor especificar cuáles argumentos son pasados por referencia únicamente en la declaración de la función.

Vea también las Referencias Explicadas.

expose_php **boolean**

Decide si PHP debe exponer el hecho de que está instalado en el servidor (p.ej. agregando su firma en la cabecera del servidor Web). No constituye un riesgo de seguridad en ninguna forma, pero hace posible determinar si usted usa PHP es su servidor o no.

zend.ze1_compatibility_mode **boolean**

Habilita el modo de compatibilidad con el Motor Zend 1 (PHP 4). Afecta el modo de clonar, moldear y comparar objetos.

Límites de Recursos

Tabla H-5. Límites de Recursos

Nombre	Predeterminado	Modificable
memory_limit	"8M"	PHP_INI_ALL

A continuación se presenta una corta explicación de las directivas de configuración.

memory_limit **integer**

Este valor define la cantidad máxima de memoria en bytes que un script puede reservar. Esto ayuda a prevenir que scripts pobremente escritos terminen consumiendo toda la memoria disponible en un servidor. Para usar esta directiva, es necesario habilitarla en tiempo de compilación. De modo que la línea de llamado a configure debe incluir: *--enable-memory-limit*. Note que es necesario definir su valor a -1 si no desea imponer un límite a su memoria.

A partir de PHP 4.3.2, y siempre que memory_limit se encuentre habilitado, la función de PHP **memory_get_usage()** estará disponible.

Cuando se usa un número entero, el valor del mismo es medido en bytes. También se puede usar la notación reducida tal como se describe en esta FAQ.

Vea también: max_execution_time.

Manejo de Datos

Tabla H-6. Opciones de Configuración de Manejo de Datos

Nombre	Predeterminado	Modificable
track_vars	"On"	PHP_INI_??
arg_separator.output	"&"	PHP_INI_ALL
arg_separator.input	"&"	PHP_INI_SYSTEM| PHP_INI_PERDIR
variables_order	"EGPCS"	PHP_INI_ALL
register_globals	"Off"	PHP_INI_PERDIR| PHP_INI_SYSTEM
register_argc_argv	"On"	PHP_INI_PERDIR| PHP_INI_SYSTEM
register_long_arrays	"On"	PHP_INI_PERDIR| PHP_INI_SYSTEM
post_max_size	"8M"	PHP_INI_SYSTEM| PHP_INI_PERDIR
gpc_order	"GPC"	PHP_INI_ALL
auto_prepend_file	""	PHP_INI_SYSTEM| PHP_INI_PERDIR

Nombre	Predeterminado	Modificable
auto_append_file	""	PHP_INI_SYSTEM| PHP_INI_PERDIR
default_mimetype	"text/html"	PHP_INI_ALL
default_charset	"iso-8859-1"	PHP_INI_ALL
always_populate_raw_post_data	"0"	PHP_INI_SYSTEM| PHP_INI_PERDIR
allow_webdav_methods	"0"	PHP_INI_SYSTEM| PHP_INI_PERDIR

A continuación se presenta una corta explicación de las directivas de configuración.

track_vars **boolean**

Si esta opción es habilitada, entonces las variables de Entorno, GET, POST, Cookies y Servidor pueden encontrarse en las matrices asociativas globales *$_ENV*, *$_GET*, *$_POST*, *$_COOKIE*, y *$_SERVER*.

Note que a partir de PHP 4.0.3, track_vars se encuentra habilitada siempre.

arg_separator.output **string**

El separador usado en URLs generadas por PHP para separar los argumentos.

arg_separator.input **string**

Lista de separadores usados por PHP para interpretar URLs de entrada en variables.

Nota: ¡Cada caracter en esta directiva es considerado como separador!

variables_order **string**

Establece el orden de procesamiento de variables EGPCS (Entorno, GET, POST, Cookie, Servidor). La configuración predeterminada de esta directiva es "EGPCS". Al definir este valor como "GP", por ejemplo, causará que PHP ignore completamente las variables de entorno, cookies y servidor, asi como que cualquier variable del método GET sea sobrescrita con una variable del método POST con el mismo nombre.

Vea también register_globals.

register_globals **boolean**

Indica si las variables EGPCS (Entorno, GET, POST, Cookie, Servidor) deben registrarse como variables globales o no.

A partir de PHP 4.2.0, esta directiva tiene el valor predeterminado *off*.

Por favor lea el capítulo de seguridad sobre el Uso de register_globals para más información al respecto.

Por favor note que register_globals no puede ser definido en tiempo de ejecución (**ini_set()**). Sin embargo, puede usar `.htaccess` si su servidor huésped lo permite como se describe

anteriormente. Un ejemplo de entrada en `.htaccess`: `php_flag register_globals off`.

Nota: El valor register_globals es afectado por la directiva variables_order.

register_argc_argv **boolean**

Le dice a PHP si declarar las variables argv y argc (que contendrían la información de GET).

Vea también la documentación sobre la línea de comandos. Asimismo, esta directiva se encuentra disponible desde PHP 4.0.0, y su valor siempre fue "on" anteriormente.

register_long_arrays **boolean**

Le dice a PHP si debe registrar o no las variables predeterminadas largas tipo *$HTTP_*_VARS*. Cuando su valor es On (predeterminado), las variables de PHP largas predefinidas como *$HTTP_GET_VARS* serán definidas. Si no las usa, se recomienda deshablitar esta opción, por razones de rendimiento. En su lugar, use las matrices superglobales, como *$_GET*.

Esta directiva se encuentra disponible a partir de PHP 5.0.0.

post_max_size **integer**

Define el tamaño máximo permitido de datos enviados mediante el método post. Este parámetro afecta también la carga de archivos. Para cargar archivos grandes, este valor debe ser mayor que upload_max_filesize.

Si el límite de memoria es habilitado mediante su script configure, memory_limit afecta también la carga de archivos. En general, memory_limit debería ser mayor que *post_max_size*.

Cuando se usa un número entero, el valor del mismo es medido en bytes. También se puede usar la notación reducida tal como se describe en esta FAQ.

gpc_order **string**

Define el orden de procesamiento de variables GET/POST/COOKIE. El valor predeterminado de esta directiva es "GPC". Al definir este valor como "GP", por ejemplo, se causará que PHP ignore por completo las cookies y sobrescriba cualquier variables proveniente del método GET con variables del método POST que tengan el mismo nombre.

Nota: Esta opción no se encuentra disponible en PHP 4. Use variables_order en su lugar.

auto_prepend_file **string**

Especifica el nombre de un archivo que es interpretado automáticamente antes del archivo principal. El archivo es incluido como si fuera llamado con la función **include()**, de modo que se usa el valor de include_path.

El valor especial none deshabilita el preprocesamiento automático.

auto_append_file **string**

Especifica el nombre de un archivo que es interpretado automáticamente después del archivo principal. El archivo es incluido como si fuera llamado con la función **include()**, así que el valor de include_path es usado.

El valor especial none deshabilita el procesamiento posterior automático.

Nota: Si el script es terminado con **exit()**, el procesamionto posterior *no* occurirá.

default_mimetype **string**

default_charset **string**

A partir de 4.0b4, PHP siempre imprime de manera predeterminada una codificación de caracteres en la cabecera Content-type:. Para deshabilitar el envío del juego de caracteres, simplemente defina este valor como vacío.

always_populate_raw_post_data **boolean**

Poblar siempre la variable $HTTP_RAW_POST_DATA.

allow_webdav_methods **boolean**

Permitir el manejo de peticiones WebDAV http al interior de scripts PHP (p.ej. PROPFIND, PROPPATCH, MOVE, COPY, etc.). Esta directiva no existe a la altura de PHP 4.3.2. Si desea obtener los datos enviados desde tales peticiones, tiene que definir always_populate_raw_post_data también.

Vea también: magic_quotes_gpc, magic_quotes_runtime, y magic_quotes_sybase.

Rutas y Directorios

Tabla H-7. Opciones de Configuración de Rutas y Directorios

Nombre	Predeterminado	Modificable
include_path	PHP_INCLUDE_PATH	PHP_INI_ALL
doc_root	PHP_INCLUDE_PATH	PHP_INI_SYSTEM
user_dir	NULL	PHP_INI_SYSTEM
extension_dir	PHP_EXTENSION_DIR	PHP_INI_SYSTEM
cgi.fix_pathinfo	"0"	PHP_INI_SYSTEM
cgi.force_redirect	"1"	PHP_INI_SYSTEM
cgi.redirect_status_env	""	PHP_INI_SYSTEM
fastcgi.impersonate	"0"	PHP_INI_SYSTEM
cgi.rfc2616_headers	"0"	PHP_INI_SYSTEM

A continuación se presenta una corta explicación de las directivas de configuración.

include_path **string**

Especifica una lista de directorios en donde las funciones **require()**, **include()** y **fopen_with_path()** buscan archivos. El formato es como aquel de la variable de entorno de sistema *PATH*: una lista de directorios separada con dos-puntos en Unix o punto-y-coma en Windows.

Ejemplo H-1. include_path en Unix

```
include_path=".:/php/includes"
```

Ejemplo H-2. include_path en Windows

```
include_path=".;c:\php\includes"
```

Mediante el uso de . en la ruta de inclusión es posible definir inclusiones relativas, ya que su valor se traduce como el directorio actual.

doc_root **string**

El "directorio raíz" de PHP en el servidor. Usado solamente si su valor no es vacío. Si PHP es configurado con safe mode, no se servirán archivos por fuera de este directorio. Si PHP no fue compilado con FORCE_REDIRECT, usted *debería* definir doc_root en caso de estar usando PHP como CGI bajo cualquier servidor web (diferente a IIS). La alternativa es usar el parámetro de configuración cgi.force_redirect descrito más adelante.

user_dir **string**

El nombre base del directorio usado en un directorio de usuario para archivos PHP, por ejemplo `public_html`.

extension_dir **string**

El directorio en donde PHP ha de buscar por extensiones cargadas dinámicamente. Vea también: enable_dl, y **dl()**.

extension **string**

Cuáles extensiones de carga dinámica leer cuando PHP inicia.

cgi.fix_pathinfo **boolean**

Ofrece un soporte *real* de PATH_INFO/PATH_TRANSLATED para CGI. El comportamiento anterior de PHP era definir PATH_TRANSLATED como SCRIPT_FILENAME, y no producir el valor PATH_INFO. Para más información sobre PATH_INFO, consulte las especificaciones sobre el estándar cgi. Definir este parámetro como 1 causará que el modo CGI de PHP fije su ruta para que cumpla con la especificación. Un valor de cero causa que PHP se comporte como lo hacía anteriormente. El valor predeterminado es cero. Es recomendable que arregle sus scripts para que usen SCRIPT_FILENAME en lugar de PATH_TRANSLATED.

cgi.force_redirect **boolean**

cgi.force_redirect es necesario para ofrecer seguridad cuando PHP es ejecutado como CGI bajo la mayoría de servidores web. Si no es definido, PHP habilita este parámetro por defecto. Es posible deshabilitarlo *bajo su propio riesgo*.

> **Nota:** Usuarios de Windows: Es *posible* deshabilitar esta opción para IIS, de hecho, es *necesario* hacerlo. Para lograr que OmniHTTPD o Xitami funcionen es

necesario deshabilitar este parámetro.

cgi.redirect_status_env **string**

Si cgi.force_redirect se encuentra habilitado, y no está usando servidores web Apache o Netscape (iPlanet), *puede* que necesite definir un nombre de variable de entorno que PHP use para saber si está bien continuar con la ejecución.

Nota: Definir esta variable *puede* causar problemas de seguridad, *asegúrese de saber lo que hace primero*.

fastcgi.impersonate **string**

FastCGI bajo IIS (en un SO basado en WINNT) soporta la habilidad de imitar tokens de seguridad del cliente que hace las peticiones. Esto permite que IIS defina el contexto de seguridad bajo el cual es ejecutada la petición. mod_fastcgi bajo apache no soporta esta característica por el momento (03/17/2002). Defina su valor como 1 si está usando IIS. Su valor predeterminado es cero.

cgi.rfc2616_headers **int**

Le dice a PHP qué tipo de cabeceras usar cuando envíe los códigos de respuesta HTTP. Si su valor es 0, PHP envía una cabecera Status: que es soportada por Apache y otros servidores web. Cuando esta opción tiene el valor de 1, PHP enviará cabeceras compatibles con el documento RFC 2616. Deje su valor en 0 a menos que sepa lo que está haciendo.

Carga de Archivos

Tabla H-8. Opciones de Configuración de Carga de Archivos

Nombre	Predeterminado	Modificable
file_uploads	"1"	PHP_INI_SYSTEM
upload_tmp_dir	NULL	PHP_INI_SYSTEM
upload_max_filesize	"2M"	PHP_INI_SYSTEM| PHP_INI_PERDIR

A continuación se presenta una corta explicación de las directivas de configuración.

file_uploads **boolean**

Indica si se permite o no la carga de archivos HTTP. Vea también las directivas upload_max_filesize, upload_tmp_dir, y post_max_size.

Cuando se usa un número entero, el valor del mismo es medido en bytes. También se puede usar la notación reducida tal como se describe en esta FAQ.

upload_tmp_dir **string**

El directorio temporal usado para almacenar archivos cuando se realiza carga de archivos. Debe tener permisos de escritura para el usuario bajo el que PHP es ejecutado. Si no se especifica, PHP usará el valor predeterminado del sistema.

upload_max_filesize **integer**

El tamaño máximo de un archivo cargado.

Cuando se usa un número entero, el valor del mismo es medido en bytes. También se puede usar la notación reducida tal como se describe en esta FAQ.

SQL General

Tabla H-9. Opciones Generales de Configuración SQL

Nombre	Predeterminado	Modificable
sql.safe_mode	"0"	PHP_INI_SYSTEM

A continuación se presenta una corta explicación de las directivas de configuración.

sql.safe_mode **boolean**

Directivas de Configuración del Depurador

Atención
Solo PHP 3 implementa un depurador predeterminado, para más información consulte Apéndice E.

debugger.host **string**

Nombre DNS o dirección IP del host usado por el depurador.

debugger.port **string**

Número de puerto usado por el depurador.

debugger.enabled **boolean**

Indica si el depurador se encuentra habilitado.

Apéndice I. Lista de alias de funciones

Aqui teneis la lista de alias de funciones. Todos loas alias están listados aqui. en general es una mala idea usar alias, ya que pueden ser modificados o renombrados, por lo que tu código no podra utilizarse. Esta lista se provee para que quien quiera actualizar su código antiguo a la sintaxis actual.

De todas maneras, algunas funciones tienen dos nombres, sin preferencia por uno u otro. (Por ejemplo, **is_int()** y **is_integer()** som válidas indistintamente)

Esta lista es consistente con PHP 4.0.6. Para obtener una lista actualizada todos los dias, pasaros por aqui.

Tabla I-1. Alias

Alias	Función principal	Extensión usada
_	**gettext()**	Gettext
add	**swfmovie_add()**	Ming (flash)
add	**swfsprite_add()**	Ming (flash)
add_root	**domxml_add_root()**	DOM XML
addaction	**swfbutton_addAction()**	Ming (flash)
addcolor	**swfdisplayitem_addColor()**	Ming (flash)
addentry	**swfgradient_addEntry()**	Ming (flash)
addfill	**swfshape_addfill()**	Ming (flash)
addshape	**swfbutton_addShape()**	Ming (flash)
addstring	**swftext_addString()**	Ming (flash)
addstring	**swftextfield_addString()**	Ming (flash)
align	**swftextfield_align()**	Ming (flash)
attributes	**domxml_attributes()**	DOM XML
children	**domxml_children()**	DOM XML
chop	**rtrim()**	Base syntax
close	**closedir()**	Base syntax
com_get	**com_propget()**	COM
com_propset	**com_propput()**	COM
com_set	**com_propput()**	COM
cv_add	**ccvs_add()**	CCVS
cv_auth	**ccvs_auth()**	CCVS
cv_command	**ccvs_command()**	CCVS
cv_count	**ccvs_count()**	CCVS
cv_delete	**ccvs_delete()**	CCVS
cv_done	**ccvs_done()**	CCVS
cv_init	**ccvs_init()**	CCVS
cv_lookup	**ccvs_lookup()**	CCVS
cv_new	**ccvs_new()**	CCVS
cv_report	**ccvs_report()**	CCVS
cv_return	**ccvs_return()**	CCVS
cv_reverse	**ccvs_reverse()**	CCVS
cv_sale	**ccvs_sale()**	CCVS
cv_status	**ccvs_status()**	CCVS
cv_textvalue	**ccvs_textvalue()**	CCVS
cv_void	**ccvs_void()**	CCVS
die	**exit()**	Miscellaneous functions

Alias	Función principal	Extensión usada
dir	**getdir()**	Base syntax
diskfreespace	**disk_free_space()**	Filesystem
domxml_getattr	**domxml_get_attribute()**	DOM XML
domxml_setattr	**domxml_set_attribute()**	DOM XML
doubleval	**floatval()**	Base syntax
drawarc	**swfshape_drawarc()**	Ming (flash)
drawcircle	**swfshape_drawcircle()**	Ming (flash)
drawcubic	**swfshape_drawcubic()**	Ming (flash)
drawcubicto	**swfshape_drawcubicto()**	Ming (flash)
drawcurve	**swfshape_drawcurve()**	Ming (flash)
drawcurveto	**swfshape_drawcurveto()**	Ming (flash)
drawglyph	**swfshape_drawglyph()**	Ming (flash)
drawline	**swfshape_drawline()**	Ming (flash)
drawlineto	**swfshape_drawlineto()**	Ming (flash)
dtd	**domxml_intdtd()**	DOM XML
dumpmem	**domxml_dumpmem()**	DOM XML
fbsql	**fbsql_db_query()**	FrontBase
fputs	**fwrite()**	Base syntax
get_attribute	**domxml_get_attribute()**	DOM XML
getascent	**swffont_getAscent()**	Ming (flash)
getascent	**swftext_getAscent()**	Ming (flash)
getattr	**domxml_get_attribute()**	DOM XML
getdescent	**swffont_getDescent()**	Ming (flash)
getdescent	**swftext_getDescent()**	Ming (flash)
getheight	**swfbitmap_getHeight()**	Ming (flash)
getleading	**swffont_getLeading()**	Ming (flash)
getleading	**swftext_getLeading()**	Ming (flash)
getshape1	**swfmorph_getShape1()**	Ming (flash)
getshape2	**swfmorph_getShape2()**	Ming (flash)
getwidth	**swfbitmap_getWidth()**	Ming (flash)
getwidth	**swffont_getWidth()**	Ming (flash)
getwidth	**swftext_getWidth()**	Ming (flash)
gzputs	**gzwrite()**	Zlib
i18n_convert	**mb_convert_encoding()**	Multi-bytes Strings
i18n_discover_encoding	**mb_detect_encoding()**	Multi-bytes Strings
i18n_http_input	**mb_http_input()**	Multi-bytes Strings
i18n_http_output	**mb_http_output()**	Multi-bytes Strings
i18n_internal_encoding	**mb_internal_encoding()**	Multi-bytes Strings

Alias	Función principal	Extensión usada
i18n_ja_jp_hantozen	**mb_convert_kana()**	Multi-bytes Strings
i18n_mime_header_decode	**mb_decode_mimeheader()**	Multi-bytes Strings
i18n_mime_header_encode	**mb_encode_mimeheader()**	Multi-bytes Strings
imap_create	**imap_createmailbox()**	IMAP
imap_fetchtext	**imap_body()**	IMAP
imap_getmailboxes	**imap_list_full()**	IMAP
imap_getsubscribed	**imap_lsub_full()**	IMAP
imap_header	**imap_headerinfo()**	IMAP
imap_listmailbox	**imap_list()**	IMAP
imap_listsubscribed	**imap_lsub()**	IMAP
imap_rename	**imap_renamemailbox()**	IMAP
imap_scan	**imap_listscan()**	IMAP
imap_scanmailbox	**imap_listscan()**	IMAP
ini_alter	**ini_set()**	Base syntax
is_double	**is_float()**	Base syntax
is_integer	**is_int()**	Base syntax
is_long	**is_int()**	Base syntax
is_real	**is_float()**	Base syntax
is_writeable	**is_writable()**	Base syntax
join	**implode()**	Base syntax
labelframe	**swfmovie_labelFrame()**	Ming (flash)
labelframe	**swfsprite_labelFrame()**	Ming (flash)
last_child	**domxml_last_child()**	DOM XML
lastchild	**domxml_last_child()**	DOM XML
ldap_close	**ldap_unbind()**	LDAP
magic_quotes_runtime	**set_magic_quotes_runtime()**	Base syntax
mbstrcut	**mb_strcut()**	Multi-bytes Strings
mbstrlen	**mb_strlen()**	Multi-bytes Strings
mbstrpos	**mb_strpos()**	Multi-bytes Strings
mbstrrpos	**mb_strrpos()**	Multi-bytes Strings
mbsubstr	**mb_substr()**	Multi-bytes Strings
ming_setcubicthreshold	**ming_setCubicThreshold()**	Ming (flash)
ming_setscale	**ming_setScale()**	Ming (flash)
move	**swfdisplayitem_move()**	Ming (flash)
movepen	**swfshape_movepen()**	Ming (flash)
movepento	**swfshape_movepento()**	Ming (flash)
moveto	**swfdisplayitem_moveTo()**	Ming (flash)
moveto	**swffill_moveTo()**	Ming (flash)

Alias	Función principal	Extensión usada
moveto	**swftext_moveTo()**	Ming (flash)
msql	**msql_db_query()**	mSQL
msql_createdb	**msql_create_db()**	mSQL
msql_dbname	**msql_result()**	mSQL
msql_dropdb	**msql_drop_db()**	mSQL
msql_fieldflags	**msql_field_flags()**	mSQL
msql_fieldlen	**msql_field_len()**	mSQL
msql_fieldname	**msql_field_name()**	mSQL
msql_fieldtable	**msql_field_table()**	mSQL
msql_fieldtype	**msql_field_type()**	mSQL
msql_freeresult	**msql_free_result()**	mSQL
msql_listdbs	**msql_list_dbs()**	mSQL
msql_listfields	**msql_list_fields()**	mSQL
msql_listtables	**msql_list_tables()**	mSQL
msql_numfields	**msql_num_fields()**	mSQL
msql_numrows	**msql_num_rows()**	mSQL
msql_regcase	**sql_regcase()**	mSQL
msql_selectdb	**msql_select_db()**	mSQL
msql_tablename	**msql_result()**	mSQL
mssql_affected_rows	**sybase_affected_rows()**	Sybase
mssql_affected_rows	**sybase_affected_rows()**	Sybase
mssql_close	**sybase_close()**	Sybase
mssql_close	**sybase_close()**	Sybase
mssql_connect	**sybase_connect()**	Sybase
mssql_connect	**sybase_connect()**	Sybase
mssql_data_seek	**sybase_data_seek()**	Sybase
mssql_data_seek	**sybase_data_seek()**	Sybase
mssql_fetch_array	**sybase_fetch_array()**	Sybase
mssql_fetch_array	**sybase_fetch_array()**	Sybase
mssql_fetch_field	**sybase_fetch_field()**	Sybase
mssql_fetch_field	**sybase_fetch_field()**	Sybase
mssql_fetch_object	**sybase_fetch_object()**	Sybase
mssql_fetch_object	**sybase_fetch_object()**	Sybase
mssql_fetch_row	**sybase_fetch_row()**	Sybase
mssql_fetch_row	**sybase_fetch_row()**	Sybase
mssql_field_seek	**sybase_field_seek()**	Sybase
mssql_field_seek	**sybase_field_seek()**	Sybase
mssql_free_result	**sybase_free_result()**	Sybase

Alias	Función principal	Extensión usada
mssql_free_result	**sybase_free_result()**	Sybase
mssql_get_last_message	**sybase_get_last_message()**	Sybase
mssql_get_last_message	**sybase_get_last_message()**	Sybase
mssql_min_client_severity	**sybase_min_client_severity()**	Sybase
mssql_min_error_severity	**sybase_min_error_severity()**	Sybase
mssql_min_message_severity	**sybase_min_message_severity()**	Sybase
mssql_min_server_severity	**sybase_min_server_severity()**	Sybase
mssql_num_fields	**sybase_num_fields()**	Sybase
mssql_num_fields	**sybase_num_fields()**	Sybase
mssql_num_rows	**sybase_num_rows()**	Sybase
mssql_num_rows	**sybase_num_rows()**	Sybase
mssql_pconnect	**sybase_pconnect()**	Sybase
mssql_pconnect	**sybase_pconnect()**	Sybase
mssql_query	**sybase_query()**	Sybase
mssql_query	**sybase_query()**	Sybase
mssql_result	**sybase_result()**	Sybase
mssql_result	**sybase_result()**	Sybase
mssql_select_db	**sybase_select_db()**	Sybase
mssql_select_db	**sybase_select_db()**	Sybase
multcolor	**swfdisplayitem_multColor()**	Ming (flash)
mysql	**mysql_db_query()**	MySQL
mysql_createdb	**mysql_create_db()**	MySQL
mysql_db_name	**mysql_result()**	MySQL
mysql_dbname	**mysql_result()**	MySQL
mysql_dropdb	**mysql_drop_db()**	MySQL
mysql_fieldflags	**mysql_field_flags()**	MySQL
mysql_fieldlen	**mysql_field_len()**	MySQL
mysql_fieldname	**mysql_field_name()**	MySQL
mysql_fieldtable	**mysql_field_table()**	MySQL
mysql_fieldtype	**mysql_field_type()**	MySQL
mysql_freeresult	**mysql_free_result()**	MySQL
mysql_listdbs	**mysql_list_dbs()**	MySQL
mysql_listfields	**mysql_list_fields()**	MySQL
mysql_listtables	**mysql_list_tables()**	MySQL
mysql_numfields	**mysql_num_fields()**	MySQL
mysql_numrows	**mysql_num_rows()**	MySQL
mysql_selectdb	**mysql_select_db()**	MySQL

Alias	Función principal	Extensión usada
mysql_tablename	**mysql_result()**	MySQL
name	**domxml_attrname()**	DOM XML
new_child	**domxml_new_child()**	DOM XML
new_xmldoc	**domxml_new_xmldoc()**	DOM XML
nextframe	**swfmovie_nextFrame()**	Ming (flash)
nextframe	**swfsprite_nextFrame()**	Ming (flash)
node	**domxml_node()**	DOM XML
oci8append	**ocicollappend()**	OCI8
oci8assign	**ocicollassign()**	OCI8
oci8assignelem	**ocicollassignelem()**	OCI8
oci8close	**ocicloselob()**	OCI8
oci8free	**ocifreecoll()**	OCI8
oci8free	**ocifreedesc()**	OCI8
oci8getelem	**ocicollgetelem()**	OCI8
oci8load	**ociloadlob()**	OCI8
oci8max	**ocicollmax()**	OCI8
oci8ocifreecursor	**ocifreestatement()**	OCI8
oci8save	**ocisavelob()**	OCI8
oci8savefile	**ocisavelobfile()**	OCI8
oci8size	**ocicollsize()**	OCI8
oci8trim	**ocicolltrim()**	OCI8
oci8writetemporary	**ociwritetemporarylob()**	OCI8
oci8writetofile	**ociwritelobtofile()**	OCI8
odbc_do	**odbc_exec()**	OCI8
odbc_field_precision	**odbc_field_len()**	OCI8
output	**swfmovie_output()**	Ming (flash)
parent	**domxml_parent()**	DOM XML
pdf_add_outline	**pdf_add_bookmark()**	PDF
pg_clientencoding	**pg_client_encoding()**	PostgreSQL
pg_setclientencoding	**pg_set_client_encoding()**	PostgreSQL
pos	**current()**	Base syntax
recode	**recode_string()**	Recode
remove	**swfmovie_remove()**	Ming (flash)
remove	**swfsprite_remove()**	Ming (flash)
rewind	**rewinddir()**	Base syntax
root	**domxml_root()**	DOM XML
rotate	**swfdisplayitem_rotate()**	Ming (flash)
rotateto	**swfdisplayitem_rotateTo()**	Ming (flash)

Alias	Función principal	Extensión usada
rotateto	**swffill_rotateTo()**	Ming (flash)
save	**swfmovie_save()**	Ming (flash)
savetofile	**swfmovie_saveToFile()**	Ming (flash)
scale	**swfdisplayitem_scale()**	Ming (flash)
scaleto	**swfdisplayitem_scaleTo()**	Ming (flash)
scaleto	**swffill_scaleTo()**	Ming (flash)
set_attribute	**domxml_set_attribute()**	DOM XML
set_content	**domxml_set_content()**	DOM XML
setaction	**swfbutton_setAction()**	Ming (flash)
setattr	**domxml_set_attribute()**	DOM XML
setbackground	**swfmovie_setBackground()**	Ming (flash)
setbounds	**swftextfield_setBounds()**	Ming (flash)
setcolor	**swftext_setColor()**	Ming (flash)
setcolor	**swftextfield_setColor()**	Ming (flash)
setdepth	**swfdisplayitem_setDepth()**	Ming (flash)
setdimension	**swfmovie_setDimension()**	Ming (flash)
setdown	**swfbutton_setDown()**	Ming (flash)
setfont	**swftext_setFont()**	Ming (flash)
setfont	**swftextfield_setFont()**	Ming (flash)
setframes	**swfmovie_setFrames()**	Ming (flash)
setframes	**swfsprite_setFrames()**	Ming (flash)
setheight	**swftext_setHeight()**	Ming (flash)
setheight	**swftextfield_setHeight()**	Ming (flash)
sethit	**swfbutton_setHit()**	Ming (flash)
setindentation	**swftextfield_setIndentation()**	Ming (flash)
setleftfill	**swfshape_setleftfill()**	Ming (flash)
setleftmargin	**swftextfield_setLeftMargin()**	Ming (flash)
setline	**swfshape_setline()**	Ming (flash)
setlinespacing	**swftextfield_setLineSpacing()**	Ming (flash)
setmargins	**swftextfield_setMargins()**	Ming (flash)
setmatrix	**swfdisplayitem_setMatrix()**	Ming (flash)
setname	**swfdisplayitem_setName()**	Ming (flash)
setname	**swftextfield_setName()**	Ming (flash)
setover	**swfbutton_setOver()**	Ming (flash)
setrate	**swfmovie_setRate()**	Ming (flash)
setratio	**swfdisplayitem_setRatio()**	Ming (flash)
setrightfill	**swfshape_setrightfill()**	Ming (flash)
setrightmargin	**swftextfield_setRightMargin()**	Ming (flash)

Alias	Función principal	Extensión usada
setspacing	**swftext_setSpacing()**	Ming (flash)
setup	**swfbutton_setUp()**	Ming (flash)
show_source	**highlight_file ()**	Base syntax
sizeof	**count()**	Base syntax
skewx	**swfdisplayitem_skewX()**	Ming (flash)
skewxto	**swfdisplayitem_skewXTo()**	Ming (flash)
skewxto	**swffill_skewXTo()**	Ming (flash)
skewy	**swfdisplayitem_skewY()**	Ming (flash)
skewyto	**swfdisplayitem_skewYTo()**	Ming (flash)
skewyto	**swffill_skewYTo()**	Ming (flash)
snmpwalkoid	**snmprealwalk()**	SNMP
strchr	**strstr()**	Base syntax
streammp3	**swfmovie_streamMp3()**	Ming (flash)
swfaction	**swfaction_init()**	Ming (flash)
swfbitmap	**swfbitmap_init()**	Ming (flash)
swfbutton	**swfbutton_init()**	Ming (flash)
swffill	**swffill_init()**	Ming (flash)
swffont	**swffont_init()**	Ming (flash)
swfgradient	**swfgradient_init()**	Ming (flash)
swfmorph	**swfmorph_init()**	Ming (flash)
swfmovie	**swfmovie_init()**	Ming (flash)
swfshape	**swfshape_init()**	Ming (flash)
swfsprite	**swfsprite_init()**	Ming (flash)
swftext	**swftext_init()**	Ming (flash)
swftextfield	**swftextfield_init()**	Ming (flash)
unlink	**domxml_unlink_node()**	DOM XML
xptr_new_context	**xpath_new_context()**	DOM XML

Apéndice J. Lista de Palabras Reservadas

El siguiente es un listado de identificadores predefinidos en PHP. Ninguno de los identificadores listados aquí debe ser usado como identificador en alguno de sus scripts. Estas listas incluyen palabras clave y nombres de variable, constantes y clases predefinidas. Estas listas no son ni exhaustivas ni completas.

Lista de Palabras Clave

Estas palabras tienen un significado especial en PHP. Algunas de ellas representan cosas que lucen como funciones, o algunas se ven como constantes, y así sucesivamente--pero no lo son, en

realidad: son construcciones del lenguaje. Usted no puede usar ninguna de las siguientes palabras como constantes, nombres de clase, nombres de funciones o métodos. Usarlas como nombres de variables está bien, generalmente, pero puede conducir a confusiones.

Tabla J-1. Palabras Clave de PHP

and	or	xor	__FILE__	exception (PHP 5)
__LINE__	**array()**	as	break	case
class	const	continue	declare	default
die()	do	**echo()**	else	elseif
empty()	enddeclare	endfor	endforeach	endif
endswitch	endwhile	**eval()**	**exit()**	extends
for	foreach	function	global	if
include()	**include_once()**	**isset()**	**list()**	new
print()	**require()**	**require_once()**	**return()**	static
switch	**unset()**	use	var	while
__FUNCTION__	__CLASS__	__METHOD__	final (PHP 5)	php_user_filter (PHP 5)
interface (PHP 5)	implements (PHP 5)	extends	public (PHP 5)	private (PHP 5)
protected (PHP 5)	abstract (PHP 5)	clone (PHP 5)	try (PHP 5)	catch (PHP 5)
throw (PHP 5)	cfunction (PHP 4 únicamente)	old_function (PHP 4 únicamente)		

Variables Predefinidas

A partir de PHP 4.1.0, el método preferido para recuperar variables externas es mediante las superglobales mencionadas más adelante. Antes de este punto, la gente recaía en register_globals o las matrices largas predefinidas en PHP (*$HTTP_*_VARS*). A partir de PHP 5.0.0, las matrices de tipo "long" de variables predefinidas, se pueden desactivar con la directiva register_long_arrays.

Variables de servidor: *$_SERVER*

Nota: Aparecieron en 4.1.0. En versiones anteriores, utilice *$HTTP_SERVER_VARS*.

$_SERVER es una matriz que contiene información tal como cabeceras, rutas y ubicaciones de scripts. Las entradas de esta matriz son creadas por el servidor web. No existen garantías de que cada servidor vaya a proveer alguno de estos valores; puede que los servidores omitan algunos, o provean otros que no se listan aquí. Hecha esta aclaración, un gran número de estas variables hacen parte de la especificación CGI 1.1, así que puede esperar que sean definidas por el servidor.

Esta es una variable 'superglobal', o global automática. Esto simplemente quiere decir que está disponible en todos los contextos a lo largo de un script. No necesita hacer **global $_SERVER;** para acceder a ella dentro de funciones o métodos, como lo hace con *$HTTP_SERVER_VARS*.

$HTTP_SERVER_VARS contiene la misma información inicial, pero no es autoglobal. (Note que *$HTTP_SERVER_VARS* y *$_SERVER* son variables diferentes y que PHP las trata como tal)

Si la directiva register_globals está definida, entonces estas variables también estarán disponibles en el contexto global del script; esto quiere decir, por separado de las matrices *$_SERVER* y *$HTTP_SERVER_VARS*. Para información relacionada, vea el capítulo de seguridad titulado Uso de Registros Globales. Estas variables globales individuales no son autoglobales.

Usted puede encontrar o no cualquiera de los siguientes elementos en $_SERVER. Note que algunos de éstos, si es que los hay, estarán disponibles (o tendrán algún significado después de todo) si se ejecuta PHP en la línea de comandos.

'*PHP_SELF*'

El nombre de archivo del script ejecutándose actualmente, relativo a la raíz de documentos. Por ejemplo, *$_SERVER['PHP_SELF']* en un script en la dirección `http://example.com/test.php/foo.bar` sería `/test.php/foo.bar`. La constante __FILE__ contiene la ruta completa y nombre del archivo actual (es decir, incluido).

Si PHP está siendo ejecutado como un procesador de línea de comandos, esta variable contiene el nombre del script a partir de PHP 4.3.0. Anteriormente no estaba disponible.

'*argv*'

Matriz de argumentos pasados al script. Cuando el script es ejecutado en la línea de comandes, ésta entrega acesso al estilo C a los parámetros de la línea de comandos. Cuando es llamado mediante el método GET, ésta contendrá la cadena de consulta (query).

'*argc*'

Contiene el número de parámetros de línea de comandos pasados al script (si se ejecuta en la línea de comandos).

'*GATEWAY_INTERFACE*'

Qué revisión de la especificación CGI está usando el servidor; i.e. '*CGI/1.1*'.

'*SERVER_NAME*'

El nombre del servidor anfitrión bajo el que está siendo ejecutado el script actual. Si el script está corriendo en un host virtual, éste será el valor definido para tal host virtual.

'*SERVER_SOFTWARE*'

Cadena de identificación del servidor, dada en las cabeceras cuando se responde a peticiones.

'*SERVER_PROTOCOL*'

Nombre y revisión del protocolo de información mediante el cual fue solicitada la página; es decir, '*HTTP/1.0*';

'*REQUEST_METHOD*'

Cuál método de petición fue usado para acceder a la página; es decir, '*GET*', '*HEAD*', '*POST*',

'*PUT*'.

'*REQUEST_TIME*'

La marca de tiempo del inicio de la petición. Disponible desde PHP 5.1.0.

'*QUERY_STRING*'

La cadena de consulta, si existe, mediante la cual se accedió a la página.

'*DOCUMENT_ROOT*'

El directorio raíz de documentos bajo el que está siendo ejecutado el script actual, tal y como se define en el archivo de configuración del servidor.

'*HTTP_ACCEPT*'

Contenidos de la cabecera *Accept:* de la petición actual, si existe.

'*HTTP_ACCEPT_CHARSET*'

Contenidos de la cabecera *Accept-Charset:* de la petición actual, si existe. Ejemplo: '*iso-8859-1,*,utf-8*'.

'*HTTP_ACCEPT_ENCODING*'

Contenidos de la cabecera *Accept-Encoding:* de la petición actual, si existe. Ejemplo: '*gzip*'.

'*HTTP_ACCEPT_LANGUAGE*'

Contenidos de la cabecera *Accept-Language:* de la petición actual, si existe. Ejemplo: '*en*'.

'*HTTP_CONNECTION*'

Contenidos de la cabecera *Connection:* de la petición actual, si existe. Ejemplo: '*Keep-Alive*'.

'*HTTP_HOST*'

Contenidos de la cabecera *Host:* de la petición actual, si existe.

'*HTTP_REFERER*'

La dirección de la página (si la hay) la cual refirió al agente de usuario a la página actual. Este valor es definido por el agente de usuario. No todos los agentes de usuario lo definen, y algunos proveen la capacidad de modificar *HTTP_REFERER* como una característica del software. En resumen, no se puede confiar realmente en este valor.

'*HTTP_USER_AGENT*'

Contenidos de la cabecera *User-Agent:* de la petición actual, si existe. Esta es una cadena que denota el agente de usuario que está accediendo a la página. Un ejemplo típico es: `Mozilla/4.5 [en] (X11; U; Linux 2.2.9 i586)`. Entre otras cosas, puede usar este valor con **get_browser()** para personalizar la salida de su página a las capacidades del agente de usuario.

'*REMOTE_ADDR*'

La dirección IP desde donde el usuario está observado la página actual.

'*REMOTE_HOST*'

El nombre Host desde donde el usuario está viendo la página actual. La consulta dns de vuelta está basada en el valor *REMOTE_ADDR* del usuario.

> **Nota:** Su servidor web debe estar configurado para crear esta variable. Por ejemplo, en Apache necesitará *HostnameLookups On* dentro de `httpd.conf` para que exista. Vea también **gethostbyaddr()**.

'*REMOTE_PORT*'

Es puerto que está siendo usado en la máquina del usuario para comunicarse con el servidor web.

'*SCRIPT_FILENAME*'

La ruta absoluta del nombre del script siendo ejecutado actualmente.

> **Nota:** Si un script es ejecutado en el entorno CLI usando una ruta relativa, tal como `archivo.php` o `../archivo.php`, *$_SERVER['SCRIPT_FILENAME']* contendrá la ruta relativa especificada por el usuario.

'*SERVER_ADMIN*'

El valor dado a la directiva SERVER_ADMIN (para Apache) en el archivo de configuración del servidor web. Si el script está siendo ejecutado en un host virtual, éste será el valor definido para ese host virtual.

'*SERVER_PORT*'

El puerto en el equipo servidor que está siendo usado por el servidor web para comunicación. En configuraciones predeterminadas, ese valor será '*80*'; usando SSL, por ejemplo, este valor cambiará a cualquiera que sea el puerto que esté definido para HTTP seguro.

'*SERVER_SIGNATURE*'

Cadena que contiene la versión del servidor y el nombre de host virtual que es agregado a las páginas generadas por el servidor, si está habilitada esta funcionalidad.

'*PATH_TRANSLATED*'

Ruta sobre el sistema de archivos (no la raíz de documentos) al script actual, luego de que el servidor haya realizado cualquier asignación al vuelo virtual-a-real.

> **Nota:** A partir de PHP 4.3.2, *PATH_TRANSLATED* ya no se define implícitamente bajo la SAPI de Apache 2, a diferencia de lo que ocurre en Apache 1, en donde se define con el mismo valor de la variable de servidor *SCRIPT_FILENAME* cuando Apache no se encarga de definirlo. Este cambio fue realizado para cumplir con la especificación CGI de que *PATH_TRANSLATED* debe existir únicamente si *PATH_INFO* se define.

Los usuarios de Apache 2 pueden usar *AcceptPathInfo* = *On* al interior de `httpd.conf` para definir *PATH_INFO*.

'*SCRIPT_NAME*'

Contiene la ruta del script actual. Ésta es útil para páginas que necesitan apuntar a ellas mismas. La constante __FILE__ contiene la ruta completa y nombre del archivo actual (es decir, incluido).

'*REQUEST_URI*'

El URI que fue dado para acceder a esta página; por ejemplo, '*/index.html*'.

'*PHP_AUTH_USER*'

Cuando se corre sobre Apache como módulo realizando autenticación HTTP, ésta variable es definida con el nombre de usuario definido por el cliente.

'*PHP_AUTH_PW*'

Cuando se corre sobre Apache como módulo realizando autenticación HTTP, ésta variable es definida con la contraseña entregada por el usuario.

'*AUTH_TYPE*'

Cuando se corre sobre Apache como módulo realizando autenticación HTTP, ésta variable es definida con el tipo de autenticación.

Variables de entorno: *$_ENV*

Nota: Introducidas en 4.1.0. En versiones anteriores, use *$HTTP_ENV_VARS*.

Estas variables son importadas en el espacio de nombres global de PHP desde el entorno bajo el que está siendo ejecutado el intérprete PHP. Muchas son entregadas por el intérprete de comandos bajo el que PHP está corriendo y diferentes sistemas suelen tener diferentes tipos de intérpretes de comandos, una lista definitiva es imposible. Por favor consulte la documentación de su intérprete de comandos por una lista de varibles de entorno que resultan definidas.

Otras variables de entorno incluyen las variables CGI, colocadas allí independientemente de que PHP esté siendo ejecutado como módulo del servidor o procesador CGI.

Esta es una variable 'superglobal', o global automática. Esto simplemente quiere decir que está disponible en todos los contextos a lo largo del script. No necesita hacer **global $_ENV;** para acceder a ella desde funciones o métodos, tal y como lo hace con *$HTTP_ENV_VARS*.

$HTTP_ENV_VARS contiene la misma información inicial, pero no es autoglobal. (Note que *$HTTP_ENV_VARS* y *$_ENV* son variables diferentes y que PHP las trata como tal)

Si la directiva register_globals está definida, entonces estas variables también se harán disponibles en el entorno global del script; i.e., por separado de las matrices *$_ENV* y *$HTTP_ENV_VARS*. Para información relacionada, consulte el capítulo de seguridad titulado Uso de Registros Globales. Estas globales individuales no son autoglobales.

Cookies HTTP: *$_COOKIE*

Nota: Introducidas en 4.1.0. En versiones anteriores, use *$HTTP_COOKIE_VARS*.

Una matriz asociativa de variables pasadas al script actual a través de cookies HTTP. Global automáticamente en cualquier contexto.

Esta es una variable 'superglobal', o global automática. Esto simplemente quiere decir que está disponible en todos los contextos a lo largo de un script. No necesita hacer **global $_COOKIE;** para acceder a ella dentro de funciones o métodos, como lo hace con *$HTTP_COOKIE_VARS*.

$HTTP_COOKIE_VARS contiene la misma información inicial, pero no es autoglobal. (Note que *$HTTP_COOKIE_VARS* y *$_COOKIE* son variables diferentes y que PHP las trata como tal)

Si la directiva register_globals está definida, entonces éstas variables también estarán disponibles en el contexto global del script; i.e., por separado de las matrices *$_COOKIE* y *$HTTP_COOKIE_VARS*. Para información relacionada, consulte el capítulo de seguridad titulado Uso de Registros Globales. Estas globales individuales no son autoglobales.

Variables HTTP GET: *$_GET*

Nota: Introducidas en 4.1.0. En versiones anteriores, use *$HTTP_GET_VARS*.

Una matriz asociativa de variables pasadas al script actual a través del método HTTP GET. Global automáticamente en cualquier contexto.

Esta es una variable 'superglobal', o global automática. Esto simplemente quiere decir que está disponible en todos los contextos a lo largo de un script. No necesita hacer **global $_GET;** para acceder a ella dentro de funciones o métodos, como lo hace con *$HTTP_GET_VARS*.

$HTTP_GET_VARS contiene la misma información inicial, pero no es autoglobal. (Note que *$HTTP_GET_VARS* y *$_GET* son variables diferentes y que PHP las trata como tal)

Si la directiva register_globals está definida, entonces éstas variables también estarán disponibles en el contexto global del script; i.e., por separado de las matrices *$_GET* y *$HTTP_GET_VARS*. Para información relacionada, consulte el capítulo de seguridad titulado Uso de Registros Globales. Estas globales individuales no son autoglobales.

Variables HTTP POST: *$_POST*

Nota: Introducidas en 4.1.0. En versiones anteriores, use *$HTTP_POST_VARS*.

Una matriz asociativa de variables pasadas al script actual a través del método HTTP POST. Global automáticamente en cualquier contexto.

Esta es una variable 'superglobal', o global automática. Esto simplemente quiere decir que está disponible en todos los contextos a lo largo de un script. No necesita hacer **global $_POST;** para acceder a ella dentro de funciones o métodos, como lo hace con *$HTTP_POST_VARS*.

$HTTP_POST_VARS contiene la misma información inicial, pero no es autoglobal. (Note que *$HTTP_POST_VARS* y *$_POST* son variables diferentes y que PHP las trata como tal)

Si la directiva register_globals está definida, entonces éstas variables también estarán disponibles en el contexto global del script; i.e., por separado de las matrices *$_POST* y *$HTTP_POST_VARS*. Para información relacionada, consulte el capítulo de seguridad titulado Uso de Registros Globales. Estas globales individuales no son autoglobales.

Variables de carga de archivos HTTP: *$_FILES*

> **Nota:** Introducidas en 4.1.0. En versiones anteriores, use *$HTTP_POST_FILES*.

Una matriz asociativa de elementos cargados al script actual a través del método HTTP POST. Global automáticamente en cualquier contexto.

Esta es una variable 'superglobal', o global automática. Esto simplemente quiere decir que está disponible en todos los contextos a lo largo de un script. No necesita hacer **global $_FILES;** para acceder a ella dentro de funciones o métodos, como lo hace con *$HTTP_POST_FILES*.

$HTTP_POST_FILES contiene la misma información inicial, pero no es autoglobal. (Note que *$HTTP_POST_FILES* y *$_FILES* son variables diferentes y que PHP las trata como tal)

Si la directiva register_globals está definida, entonces éstas variables también estarán disponibles en el contexto global del script; i.e., por separado de las matrices *$_FILES* y *$HTTP_POST_FILES*. Para información relacionada, consulte el capítulo de seguridad titulado Uso de Registros Globales. Estas globales individuales no son autoglobales.

Variables de petición: *$_REQUEST*

> **Nota:** Introducidas en 4.1.0. No existe una matriz ñalente en versiones anteriores.
>
> **Nota:** Antes de PHP 4.3.0, la información de *$_FILES* también era incluida en *$_REQUEST*.

Una matriz asociativa que consiste en los contenidos de *$_GET*, *$_POST*, y *$_COOKIE*.

Esta es una variable 'superglobal', o global automática. Esto simplemente quiere decir que está disponible en todos los contextos a lo largo de un script. No necesita hacer **global $_REQUEST;** para acceder a ella dentro de funciones o métodos.

Si la directiva register_globals está definida, entonces éstas variables también estarán disponibles en el contexto global del script; i.e., por separado de la matriz *$_REQUEST*. Para información relacionada, consulte el capítulo de seguridad titulado Uso de Registros Globales. Estas globales individuales no son autoglobales.

Variables de sesión: *$_SESSION*

> **Nota:** Introducidas en 4.1.0. En versiones anteriores, use *$HTTP_SESSION_VARS*.

Una matriz asociativa que contiene las variables de sesión disponibles en el script actual. Consulte la documentación sobre Funciones de Sesión para más información sobre cómo es usada ésta matriz.

Esta es una variable 'superglobal', o global automática. Esto simplemente quiere decir que está disponible en todos los contextos a lo largo de un script. No necesita hacer **global $_SESSION;** para acceder a ella dentro de funciones o métodos, como lo hace con *$HTTP_SESSION_VARS*.

$HTTP_SESSION_VARS contiene la misma información, pero no es autoglobal. (Note que *$HTTP_SESSION_VARS* y *$_SESSION* son variable diferentes y que PHP las trata como tal)

Si la directiva register_globals está definida, entonces éstas variables también estarán disponibles en el contexto global del script; i.e., por separado de las matrices *$_SESSION* y *$HTTP_SESSION_VARS*. Para información relacionada, consulte el capítulo de seguridad titulado Uso de Registros Globales. Estas globales individuales no son autoglobales.

Variables globales: *$GLOBALS*

> **Nota:** *$GLOBALS* ha estado disponible desde PHP 3.0.0.

Una matriz asociativa que contiene referencias a todas las variables que están definidas actualmente en el contexto global del script. Los nombres de las variables son las claves de la matriz.

Esta es una variable 'superglobal', o global automática. Esto simplemente quiere decir que está disponible en todos los contextos a lo largo de un script. No necesita hacer **global $_GLOBALS;** para acceder a ella dentro de funciones o métodos.

El mensaje de error previo: *$php_errormsg*

$php_errormsg es una variable que contiene el texto del último mensaje de error generado por PHP. Esta variable solo estará disponibles dentro del contexto en el que el error ocurrió, y solo si la opción de configuración track_errors está habilitada (por defecto está definida como off).

Clases Predefinidas

Clases Estándar Predefinidas

Estas clases están definidas en el juego de funciones estándar incluidas en la distribuición de PHP.

Directory

> La clase a partir de la cual dir es instanciada.

stdClass

__PHP_Incomplete_Class

Clases predefinidas en PHP 5

Estas clases predefinidas adicionales fueron introducidas en PHP 5.0.0

exception

php_user_filter

Clases Ming Definidas

Estas clases están definidas en la extensión Ming, y solo estarán disponibles cuando tal extensión haya sido compilada junto con PHP o cargada dinámicamente en tiempo de ejecución.

swfshape

swffill

swfgradient

swfbitmap

swftext

swftextfield

swffont

swfdisplayitem

swfmovie

swfbutton

swfaction

swfmorph

swfsprite

Clases Oracle 8 Definidas

Estas clases están definidas en la extensión Oracle 8, y solo estarán disponibles cuando tal extensión haya sido compilada junto con PHP o cargada dinámicamente en tiempo de ejecución.

OCI-Lob

OCI-Collection

Clases qtdom Definidas

Estas clases están definidas en la extensión qtdom, y solo estarán disponibles cuando tal extensión haya sido compilada junto con PHP o cargada dinámicamente en tiempo de ejecución.

QDomDocument

QDomNode

Constantes predefinidas

Tabla de contenidos

Constantes Base Predefinidas

Constantes Base Predefinidas -- Constantes definidas en el núcleo de PHP, Zend, y módulos SAPI

Descripción

Estas constantes son definidas por el núcleo de PHP. Esto incluye PHP, el motor Zend, y los módulos SAPI.

PHP_VERSION (**string**)

PHP_OS (**string**)

PHP_EOL (**string**)

Disponible a partir de PHP 4.3.10 y PHP 5.0.2

DEFAULT_INCLUDE_PATH (**string**)

PEAR_INSTALL_DIR (**string**)

PEAR_EXTENSION_DIR (**string**)

PHP_EXTENSION_DIR (**string**)

PHP_BINDIR (**string**)

PHP_LIBDIR (**string**)

PHP_DATADIR (**string**)

PHP_SYSCONFDIR (**string**)

PHP_LOCALSTATEDIR (**string**)

PHP_CONFIG_FILE_PATH (string)

PHP_OUTPUT_HANDLER_START (integer)

PHP_OUTPUT_HANDLER_CONT (integer)

PHP_OUTPUT_HANDLER_END (integer)

E_ERROR (integer)

E_WARNING (integer)

E_PARSE (integer)

E_NOTICE (integer)

E_CORE_ERROR (integer)

E_CORE_WARNING (integer)

E_COMPILE_ERROR (integer)

E_COMPILE_WARNING (integer)

E_USER_ERROR (integer)

E_USER_WARNING (integer)

E_USER_NOTICE (integer)

E_ALL (integer)

E_STRICT (integer)

Disponible a partir de PHP 5.0.0

Vea también: Constantes mágicas.

Constantes Estándar Predefinidas

Constantes Estándar Predefinidas -- Constantes definidas en PHP por defecto

Descripción

Estas constantes están definidas en PHP por defecto.

EXTR_OVERWRITE (integer)

EXTR_SKIP (integer)

EXTR_PREFIX_SAME (integer)

EXTR_PREFIX_ALL (integer)

EXTR_PREFIX_INVALID (integer)

EXTR_PREFIX_IF_EXISTS (integer)

EXTR_IF_EXISTS (integer)

SORT_ASC (integer)

SORT_DESC (integer)

SORT_REGULAR (integer)

SORT_NUMERIC (integer)

SORT_STRING (integer)

CASE_LOWER (integer)

CASE_UPPER (integer)

COUNT_NORMAL (integer)

COUNT_RECURSIVE (integer)

ASSERT_ACTIVE (integer)

ASSERT_CALLBACK (integer)

ASSERT_BAIL (integer)

ASSERT_WARNING (integer)

ASSERT_QUIET_EVAL (integer)

CONNECTION_ABORTED (integer)

CONNECTION_NORMAL (integer)

CONNECTION_TIMEOUT (integer)

INI_USER (integer)

INI_PERDIR (integer)

INI_SYSTEM (integer)

INI_ALL (integer)

M_E (float)

M_LOG2E (float)

M_LOG10E (float)

M_LN2 (float)

M_LN10 (float)

M_PI (float)

M_PI_2 (float)

M_PI_4 (float)

M_1_PI (float)

M_2_PI (float)

M_2_SQRTPI (float)

M_SQRT2 (float)

M_SQRT1_2 (float)

CRYPT_SALT_LENGTH (**integer**)

CRYPT_STD_DES (**integer**)

CRYPT_EXT_DES (**integer**)

CRYPT_MD5 (**integer**)

CRYPT_BLOWFISH (**integer**)

DIRECTORY_SEPARATOR (**string**)

SEEK_SET (**integer**)

SEEK_CUR (**integer**)

SEEK_END (**integer**)

LOCK_SH (**integer**)

LOCK_EX (**integer**)

LOCK_UN (**integer**)

LOCK_NB (**integer**)

HTML_SPECIALCHARS (**integer**)

HTML_ENTITIES (integer)

ENT_COMPAT (integer)

ENT_QUOTES (integer)

ENT_NOQUOTES (integer)

INFO_GENERAL (integer)

INFO_CREDITS (integer)

INFO_CONFIGURATION (integer)

INFO_MODULES (integer)

INFO_ENVIRONMENT (integer)

INFO_VARIABLES (integer)

INFO_LICENSE (integer)

INFO_ALL (integer)

CREDITS_GROUP (integer)

CREDITS_GENERAL (integer)

CREDITS_SAPI (integer)

CREDITS_MODULES (integer)

CREDITS_DOCS (integer)

CREDITS_FULLPAGE (integer)

CREDITS_QA (integer)

CREDITS_ALL (integer)

STR_PAD_LEFT (integer)

STR_PAD_RIGHT (integer)

STR_PAD_BOTH (integer)

PATHINFO_DIRNAME (integer)

PATHINFO_BASENAME (integer)

PATHINFO_EXTENSION (integer)

PATH_SEPARATOR (string)

CHAR_MAX (integer)

LC_CTYPE (integer)

LC_NUMERIC (integer)

LC_TIME (integer)

LC_COLLATE (integer)

LC_MONETARY (integer)

LC_ALL (integer)

LC_MESSAGES (integer)

ABDAY_1 (integer)

ABDAY_2 (integer)

ABDAY_3 (integer)

ABDAY_4 (integer)

ABDAY_5 (integer)

ABDAY_6 (integer)

ABDAY_7 (integer)

DAY_1 (integer)

DAY_2 (integer)

DAY_3 (integer)

DAY_4 (integer)

DAY_5 (integer)

DAY_6 (integer)

DAY_7 (integer)

ABMON_1 (integer)

ABMON_2 (integer)

ABMON_3 (integer)

ABMON_4 (integer)

ABMON_5 (integer)

ABMON_6 (integer)

ABMON_7 (integer)

ABMON_8 (integer)

ABMON_9 (integer)

ABMON_10 (integer)

ABMON_11 (integer)

ABMON_12 (integer)

MON_1 (integer)

MON_2 (integer)

MON_3 (integer)

MON_4 (integer)

MON_5 (integer)

MON_6 (integer)

MON_7 (integer)

MON_8 (integer)

MON_9 (integer)

MON_10 (integer)

MON_11 (integer)

MON_12 (integer)

AM_STR (integer)

PM_STR (integer)

D_T_FMT (integer)

D_FMT (integer)

T_FMT (integer)

T_FMT_AMPM (integer)

ERA (integer)

ERA_YEAR (integer)

ERA_D_T_FMT (integer)

ERA_D_FMT (integer)

ERA_T_FMT (integer)

ALT_DIGITS (integer)

INT_CURR_SYMBOL (integer)

CURRENCY_SYMBOL (integer)

CRNCYSTR (integer)

MON_DECIMAL_POINT (integer)

MON_THOUSANDS_SEP (integer)

MON_GROUPING (integer)

POSITIVE_SIGN (integer)

NEGATIVE_SIGN (integer)

INT_FRAC_DIGITS (integer)

FRAC_DIGITS (integer)

P_CS_PRECEDES (integer)

P_SEP_BY_SPACE (integer)

N_CS_PRECEDES (integer)

N_SEP_BY_SPACE (integer)

P_SIGN_POSN (integer)

N_SIGN_POSN (integer)

DECIMAL_POINT (integer)

RADIXCHAR (integer)

THOUSANDS_SEP (integer)

THOUSEP (integer)

GROUPING (integer)

YESEXPR (integer)

NOEXPR (integer)

YESSTR (integer)

NOSTR (integer)

CODESET (integer)

LOG_EMERG (integer)

LOG_ALERT (integer)

LOG_CRIT (integer)

LOG_ERR (integer)

LOG_WARNING (integer)

LOG_NOTICE (integer)

LOG_INFO (integer)

LOG_DEBUG (integer)

LOG_KERN (integer)

LOG_USER (integer)

LOG_MAIL (integer)

LOG_DAEMON (integer)

LOG_AUTH (integer)

LOG_SYSLOG (integer)

LOG_LPR (integer)

LOG_NEWS (integer)

LOG_UUCP (integer)

LOG_CRON (integer)

LOG_AUTHPRIV (integer)

LOG_LOCAL0 (integer)

LOG_LOCAL1 (integer)

LOG_LOCAL2 (integer)

LOG_LOCAL3 (integer)

LOG_LOCAL4 (integer)

LOG_LOCAL5 (integer)

LOG_LOCAL6 (integer)

LOG_LOCAL7 (integer)

LOG_PID (integer)

LOG_CONS (integer)

LOG_ODELAY (integer)

LOG_NDELAY (integer)

LOG_NOWAIT (integer)

LOG_PERROR (integer)

Apéndice K. Lista de Tipos de Recurso

La siguiente es una lista de las funciones que crean, usan o destruyen recursos PHP. La función **is_resource()** puede ser usada para determinar si una variable es un recurso y **get_resource_type()** devolverá el tipo de recurso que es.

Tabla K-1. Tipos de Recurso

Nombre del Tipo de Recurso	Creado Por	Usado Por	Destruido Por	Definición
aspell	**aspell_new()**	**aspell_check()**, **aspell_check_raw()**, **aspell_suggest()**	Nadie	Diccionario aspell
bzip2	**bzopen()**	**bzerrno()**, **bzerror()**, **bzerrstr()**, **bzflush()**, **bzread()**, **bzwrite()**	**bzclose()**	Archivo bzip2

Nombre del Tipo de Recurso	Creado Por	Usado Por	Destruido Por	Definición
COM	com_load()	com_invoke(), com_propget(), com_get(), com_propput(), com_set(), com_propput()	Nadie	Referencia a objeto COM
VARIANT				

Nombre del Tipo de Recurso	Creado Por	Usado Por	Destruido Por	Definición
cpdf	cpdf_open()	cpdf_page_init(), cpdf_finalize_page(), cpdf_finalize(), cpdf_output_buffer(), cpdf_save_to_file(), cpdf_set_current_page(), cpdf_begin_text(), cpdf_end_text(), cpdf_show(), cpdf_show_xy(), cpdf_text(), cpdf_set_font(), cpdf_set_leading(), cpdf_set_text_rendering(), cpdf_set_horiz_scaling(), cpdf_set_text_rise(), cpdf_set_text_matrix(), cpdf_set_text_pos(), cpdf_set_text_pos(), cpdf_set_word_spacing(), cpdf_continue_text(), cpdf_stringwidth(), cpdf_save(), cpdf_translate(), cpdf_restore(), cpdf_scale(), cpdf_rotate(), cpdf_setflat(), cpdf_setlinejoin(), cpdf_setlinecap(), cpdf_setmiterlimit(), cpdf_setlinewidth(), cpdf_setdash(), cpdf_moveto(), cpdf_rmoveto(), cpdf_curveto(), cpdf_lineto(), cpdf_rlineto(), cpdf_circle(), cpdf_arc(), cpdf_rect(), cpdf_closepath(), cpdf_stroke(), cpdf_closepath_fill_stroke(), cpdf_fill_stroke(), cpdf_clip(), cpdf_fill(), cpdf_setgray_fill(), cpdf_setgray_stroke(), cpdf_setgray(), cpdf_setrgbcolor_fill(), cpdf_setrgbcolor_stroke(), cpdf_setrgbcolor(),	cpdf_close()	Documento PDF con la biblioteca CPDF

Nombre del Tipo de Recurso	Creado Por	Usado Por	Destruido Por	Definición
cpdf outline				
curl	curl_init()	curl_init(), curl_exec()	curl_close()	Sesión curl
dbm	dbmopen()	dbmexists(), dbmfetch(), dbminsert(), dbmreplace(), dbmdelete(), dbmfirstkey(), dbmnextkey()	dbmclose()	Enlace a base de datos DBM
dba	dba_open()	dba_delete(), dba_exists(), dba_fetch(), dba_firstkey(), dba_insert(), dba_nextkey(), dba_optimize(), dba_replace(), dba_sync()	dba_close()	Enlace a base de datos DBA
dba persistent	dba_popen()	dba_delete(), dba_exists(), dba_fetch(), dba_firstkey(), dba_insert(), dba_nextkey(), dba_optimize(), dba_replace(), dba_sync()	Nadie	Enlace persistente a base de datos DBA
dbase	dbase_open()	dbase_pack(), dbase_add_record(), dbase_replace_record(), dbase_delete_record(), dbase_get_record(), dbase_get_record_with_names(), dbase_numfields(), dbase_numrecords()	dbase_close()	Enlace a base de datos Dbase
dbx_link_object	dbx_connect()	dbx_query()	dbx_close()	dbx connection
dbx_result_object	dbx_query()		Nadie	Resultado dbx
domxml attribute				
domxml document				
domxml node				
xpath context				
xpath object				
fbsql database	fbsql_select_db()		Nadie	fbsql database

Nombre del Tipo de Recurso	Creado Por	Usado Por	Destruido Por	Definición
fbsql link	fbsql_change_user(), fbsql_connect()	fbsql_autocommit(), fbsql_change_user(), fbsql_create_db(), fbsql_data_seek(), fbsql_db_query(), fbsql_drop_db(), fbsql_select_db(), fbsql_errno(), fbsql_error(), fbsql_insert_id(), fbsql_list_dbs()	fbsql_close()	Enlace a base de datos fbsql
fbsql plink	fbsql_change_user(), fbsql_pconnect()	fbsql_autocommit(), fbsql_change_user(), fbsql_create_db(), fbsql_data_seek(), fbsql_db_query(), fbsql_drop_db(), fbsql_select_db(), fbsql_errno(), fbsql_error(), fbsql_insert_id(), fbsql_list_dbs()	Nadie	Enlace persistente con base de datos fbsql
fbsql result	fbsql_db_query(), fbsql_list_dbs(), fbsql_query(), fbsql_list_fields(), fbsql_list_tables(), fbsql_tablename()	fbsql_affected_rows(), fbsql_fetch_array(), fbsql_fetch_assoc(), fbsql_fetch_field(), fbsql_fetch_lengths(), fbsql_fetch_object(), fbsql_fetch_row(), fbsql_field_flags(), fbsql_field_name(), fbsql_field_len(), fbsql_field_seek(), fbsql_field_table(), fbsql_field_type(), fbsql_next_result(), fbsql_num_fields(), fbsql_num_rows(), fbsql_result(), fbsql_num_rows()	fbsql_free_result()	fbsql result

Nombre del Tipo de Recurso	Creado Por	Usado Por	Destruido Por	Definición
fdf	fdf_open()	fdf_create(), fdf_save(), fdf_get_value(), fdf_set_value(), fdf_next_field_name(), fdf_set_ap(), fdf_set_status(), fdf_get_status(), fdf_set_file(), fdf_get_file(), fdf_set_flags(), fdf_set_opt(), fdf_set_submit_form_action(), fdf_set_javascript_action()	fdf_close()	Archivo FDF
ftp	ftp_connect()	ftp_login(), ftp_pwd(), ftp_cdup(), ftp_chdir(), ftp_mkdir(), ftp_rmdir(), ftp_nlist(), ftp_rawlist(), ftp_systype(), ftp_pasv(), ftp_get(), ftp_fget(), ftp_put(), ftp_fput(), ftp_size(), ftp_mdtm(), ftp_rename(), ftp_delete(), ftp_site()	ftp_quit()	Secuencia FTP

Nombre del Tipo de Recurso	Creado Por	Usado Por	Destruido Por	Definición
gd	imagecreate(), imagecreatefromgif(), imagecreatefromjpeg(), imagecreatefrompng(), imagecreatefromwbmp(), imagecreatefromstring(), imagecreatetruecolor()	imagearc(), imagechar(), imagecharup(), imagecolorallocate(), imagecolorat(), imagecolorclosest(), imagecolorexact(), imagecolorresolve(), imagegammacorrect(), imagegammacorrect(), imagecolorset(), imagecolorsforindex(), imagecolorstotal(), imagecolortransparent(), imagecopy(), imagecopyresized(), imagedashedline(), imagefill(), imagefilledpolygon(), imagefilledrectangle(), imagefilltoborder(), imagegif(), imagepng(), imagejpeg(), imagewbmp(), imageinterlace(), imageline(), imagepolygon(), imagepstext(), imagerectangle(), imagesetpixel(), imagestring(), imagestringup(), imagesx(), imagesy(), imagettftext(), imagefilledarc(), imageellipse(), imagefilledellipse(), imagecolorclosestalpha(), imagecolorexactalpha(), imagecolorresolvealpha(), imagecopymerge(), imagecopymergegray(), imagecopyresampled(), imagetruecolortopalette(), imagesetbrush(), imagesettile(), imagesetthickness()	imagedestroy()	Imagen GD
gd font	imageloadfont()	imagechar(), imagecharup(), imagefontheight()	Nadie	Fuente para GD

Nombre del Tipo de Recurso	Creado Por	Usado Por	Destruido Por	Definición
gd PS encoding				
gd PS font	**imagepsloadfont()**	**imagepstext()**, **imagepsslantfont()**, **imagepsextendfont()**, **imagepsencodefont()**, **imagepsbbox()**	**imagepsfreefont()**	Fuente PS para GD
GMP integer	**gmp_init()**	**gmp_intval()**, **gmp_strval()**, **gmp_add()**, **gmp_sub()**, **gmp_mul()**, **gmp_div_q()**, **gmp_div_r()**, **gmp_div_qr()**, **gmp_div()**, **gmp_mod()**, **gmp_divexact()**, **gmp_cmp()**, **gmp_neg()**, **gmp_abs()**, **gmp_sign()**, **gmp_fact()**, **gmp_sqrt()**, **gmp_sqrtrm()**, **gmp_perfect_square()**, **gmp_pow()**, **gmp_powm()**, **gmp_prob_prime()**, **gmp_gcd()**, **gmp_gcdext()**, **gmp_invert()**, **gmp_legendre()**, **gmp_jacobi()**, **gmp_random()**, **gmp_and()**, **gmp_or()**, **gmp_xor()**, **gmp_setbit()**, **gmp_clrbit()**, **gmp_scan0()**, **gmp_scan1()**, **gmp_popcount()**, **gmp_hamdist()**	Nadie	Número GMP

Nombre del Tipo de Recurso	Creado Por	Usado Por	Destruido Por	Definición
hyperwave document	**hw_cp()**, **hw_docbyanchor()**, **hw_getremote()**, **hw_getremotechildren()**	**hw_children()**, **hw_childrenobj()**, **hw_getparents()**, **hw_getparentsobj()**, **hw_getchildcoll()**, **hw_getchildcollobj()**, **hw_getremote()**, **hw_getsrcbydestobj()**, **hw_getandlock()**, **hw_gettext()**, **hw_getobjectbyquerycoll()**, **hw_getobjectbyquerycollobj()**, **hw_getchilddoccoll()**, **hw_getchilddoccollobj()**, **hw_getanchors()**, **hw_getanchorsobj()**, **hw_inscoll()**, **hw_pipedocument()**, **hw_unlock()**	**hw_deleteobject()**	Objeto hyperwave

Nombre del Tipo de Recurso	Creado Por	Usado Por	Destruido Por	Definición
hyperwave link	hw_connect()	hw_children(), hw_childrenobj(), hw_cp(), hw_deleteobject(), hw_docbyanchor(), hw_docbyanchorobj(), hw_errormsg(), hw_edittext(), hw_error(), hw_getparents(), hw_getparentsobj(), hw_getchildcoll(), hw_getchildcollobj(), hw_getremote(), hw_getremotechildren(), hw_getsrcbydestobj(), hw_getobject(), hw_getandlock(), hw_gettext(), hw_getobjectbyquery(), hw_getobjectbyqueryobj(), hw_getobjectbyquerycoll(), hw_getobjectbyquerycollobj(), hw_getchilddoccoll(), hw_getchilddoccollobj(), hw_getanchors(), hw_getanchorsobj(), hw_mv(), hw_incollections(), hw_info(), hw_inscoll(), hw_insdoc(), hw_insertdocument(), hw_insertobject(), hw_mapid(), hw_modifyobject(), hw_pipedocument(), hw_unlock(), hw_who(), hw_getusername()	hw_close(), hw_free_document()	Enlace con servidor Hyperwave

Nombre del Tipo de Recurso	Creado Por	Usado Por	Destruido Por	Definición
hyperwave link persistent	hw_pconnect()	hw_children(), hw_childrenobj(), hw_cp(), hw_deleteobject(), hw_docbyanchor(), hw_docbyanchorobj(), hw_errormsg(), hw_edittext(), hw_error(), hw_getparents(), hw_getparentsobj(), hw_getchildcoll(), hw_getchildcollobj(), hw_getremote(), hw_getremotechildren(), hw_getsrcbydestobj(), hw_getobject(), hw_getandlock(), hw_gettext(), hw_getobjectbyquery(), hw_getobjectbyqueryobj(), hw_getobjectbyquerycoll(), hw_getobjectbyquerycollobj(), hw_getchilddoccoll(), hw_getchilddoccollobj(), hw_getanchors(), hw_getanchorsobj(), hw_mv(), hw_incollections(), hw_info(), hw_inscoll(), hw_insdoc(), hw_insertdocument(), hw_insertobject(), hw_mapid(), hw_modifyobject(), hw_pipedocument(), hw_unlock(), hw_who(), hw_getusername()	Nadie	Enlace persistente con servidor Hyperwave
icap	icap_open()	icap_fetch_event(), icap_list_events(), icap_store_event(), icap_snooze(), icap_list_alarms(), icap_delete_event()	icap_close()	Enlace con servidor icap

Nombre del Tipo de Recurso	Creado Por	Usado Por	Destruido Por	Definición
imap	**imap_open()**	**imap_append()**, **imap_body()**, **imap_check()**, **imap_createmailbox()**, **imap_delete()**, **imap_deletemailbox()**, **imap_expunge()**, **imap_fetchbody()**, **imap_fetchstructure()**, **imap_headerinfo()**, **imap_header()**, **imap_headers()**, **imap_listmailbox()**, **imap_getmailboxes()**, **imap_get_quota()**, **imap_status()**, **imap_listsubscribed()**, **imap_set_quota()**, **imap_set_quota()**, **imap_getsubscribed()**, **imap_mail_copy()**, **imap_mail_move()**, **imap_num_msg()**, **imap_num_recent()**, **imap_ping()**, **imap_renamemailbox()**, **imap_reopen()**, **imap_subscribe()**, **imap_undelete()**, **imap_unsubscribe()**, **imap_scanmailbox()**, **imap_mailboxmsginfo()**, **imap_fetchheader()**, **imap_uid()**, **imap_msgno()**, **imap_search()**, **imap_fetch_overview()**	**imap_close()**	Enlace con servidor IMAP, POP3
imap chain persistent				
imap persistent				

Nombre del Tipo de Recurso	Creado Por	Usado Por	Destruido Por	Definición
ingres	ingres_connect()	ingres_query(), ingres_num_rows(), ingres_num_fields(), ingres_field_name(), ingres_field_type(), ingres_field_nullable(), ingres_field_length(), ingres_field_precision(), ingres_field_scale(), ingres_fetch_array(), ingres_fetch_row(), ingres_fetch_object(), ingres_rollback(), ingres_commit(), ingres_autocommit()	ingres_close()	Enlace con base ingresII
ingres persistent	ingres_pconnect()	ingres_query(), ingres_num_rows(), ingres_num_fields(), ingres_field_name(), ingres_field_type(), ingres_field_nullable(), ingres_field_length(), ingres_field_precision(), ingres_field_scale(), ingres_fetch_array(), ingres_fetch_row(), ingres_fetch_object(), ingres_rollback(), ingres_commit(), ingres_autocommit()	Nadie	Enlace persistente con base ingresII
interbase blob				
interbase link	ibase_connect()	ibase_query(), ibase_prepare(), ibase_trans()	ibase_close()	Enlace con base de datos Interbase
interbase link persistent	ibase_pconnect()	ibase_query(), ibase_prepare(), ibase_trans()	Nadie	Enlace persistente con base de datos Interbase
interbase query	ibase_prepare()	ibase_execute()	ibase_free_query()	Consulta Interbase
interbase result	ibase_query()	ibase_fetch_row(), ibase_fetch_object(), ibase_field_info(), ibase_num_fields()	ibase_free_result()	Resultado Interbase
interbase transaction	ibase_trans()	ibase_commit()	ibase_rollback()	Transacción Interbase

Nombre del Tipo de Recurso	Creado Por	Usado Por	Destruido Por	Definición
java				
ldap link	ldap_connect(), ldap_search()	ldap_count_entries(), ldap_first_attribute(), ldap_first_entry(), ldap_get_attributes(), ldap_get_dn(), ldap_get_entries(), ldap_get_values(), ldap_get_values_len(), ldap_next_attribute(), ldap_next_entry()	ldap_close()	Conexión ldap
ldap result	ldap_read()	ldap_add(), ldap_compare(), ldap_bind(), ldap_count_entries(), ldap_delete(), ldap_errno(), ldap_error(), ldap_first_attribute(), ldap_first_entry(), ldap_get_attributes(), ldap_get_dn(), ldap_get_entries(), ldap_get_values(), ldap_get_values_len(), ldap_get_option(), ldap_list(), ldap_modify(), ldap_mod_add(), ldap_mod_replace(), ldap_next_attribute(), ldap_next_entry(), ldap_mod_del(), ldap_set_option(), ldap_unbind()	ldap_free_result()	Resultado de búsqueda ldap
ldap result entry				

Nombre del Tipo de Recurso	Creado Por	Usado Por	Destruido Por	Definición
mcal	mcal_open(), mcal_popen()	mcal_create_calendar(), mcal_rename_calendar(), mcal_rename_calendar(), mcal_delete_calendar(), mcal_fetch_event(), mcal_list_events(), mcal_append_event(), mcal_store_event(), mcal_delete_event(), mcal_list_alarms(), mcal_event_init(), mcal_event_set_category(), mcal_event_set_title(), mcal_event_set_description(), mcal_event_set_start(), mcal_event_set_end(), mcal_event_set_alarm(), mcal_event_set_class(), mcal_next_recurrence(), mcal_event_set_recur_none(), mcal_event_set_recur_daily(), mcal_event_set_recur_weekly(), mcal_event_set_recur_monthly_mday(), mcal_event_set_recur_monthly_wday(), mcal_event_set_recur_yearly(), mcal_fetch_current_stream_event(), mcal_event_add_attribute(), mcal_expunge()	mcal_close()	Enlace a servidor de calendario
SWFAction				
SWFBitmap				
SWFButton				
SWFDisplayItem				
SWFFill				
SWFFont				
SWFGradient				
SWFMorph				
SWFMovie				

Nombre del Tipo de Recurso	Creado Por	Usado Por	Destruido Por	Definición
SWFShape				
SWFSprite				
SWFText				
SWFTextField				
mnogosearch agent				
mnogosearch result				
msql link	**msql_connect()**	**msql()**, **msql_create_db()**, **msql_createdb()**, **msql_drop_db()**, **msql_drop_db()**, **msql_select_db()**, **msql_select_db()**	**msql_close()**	Enlace a base de datos mSQL
msql link persistent	**msql_pconnect()**	**msql()**, **msql_create_db()**, **msql_createdb()**, **msql_drop_db()**, **msql_drop_db()**, **msql_select_db()**, **msql_select_db()**	Nadie	Enlace persistente con mSQL
msql query	**msql_query()**	**msql()**, **msql_affected_rows()**, **msql_data_seek()**, **msql_dbname()**, **msql_fetch_array()**, **msql_fetch_field()**, **msql_fetch_object()**, **msql_fetch_row()**, **msql_fieldname()**, **msql_field_seek()**, **msql_fieldtable()**, **msql_fieldtype()**, **msql_fieldflags()**, **msql_fieldlen()**, **msql_num_fields()**, **msql_num_rows()**, **msql_numfields()**, **msql_numrows()**, **msql_result()**	**msql_free_result()**, **msql_free_result()**	Resultado mSQL
mssql link	**mssql_connect()**	**mssql_query()**, **mssql_select_db()**	**mssql_close()**	Enlace con base de datos Microsoft SQL Server

Nombre del Tipo de Recurso	Creado Por	Usado Por	Destruido Por	Definición
mssql link persistent	**mssql_pconnect()**	**mssql_query()**, **mssql_select_db()**	Nadie	Enlace persistente con Microsoft SQL Server
mssql result	**mssql_query()**	**mssql_data_seek()**, **mssql_fetch_array()**, **mssql_fetch_field()**, **mssql_fetch_object()**, **mssql_fetch_row()**, **mssql_field_length()**, **mssql_field_name()**, **mssql_field_seek()**, **mssql_field_type()**, **mssql_num_fields()**, **mssql_num_rows()**, **mssql_result()**	**mssql_free_result()**	Rasultado de Microsoft SQL Server
mysql link	**mysql_connect()**	**mysql_affected_rows()**, **mysql_change_user()**, **mysql_create_db()**, **mysql_data_seek()**, **mysql_db_name()**, **mysql_db_query()**, **mysql_drop_db()**, **mysql_errno()**, **mysql_error()**, **mysql_insert_id()**, **mysql_list_dbs()**, **mysql_list_fields()**, **mysql_list_tables()**, **mysql_query()**, **mysql_result()**, **mysql_select_db()**, **mysql_tablename()**, **mysql_get_host_info()**, **mysql_get_proto_info()**, **mysql_get_server_info()**	**mysql_close()**	Enlace con base de datos MySQL

Nombre del Tipo de Recurso	Creado Por	Usado Por	Destruido Por	Definición
mysql link persistent	mysql_pconnect()	mysql_affected_rows(), mysql_change_user(), mysql_create_db(), mysql_data_seek(), mysql_db_name(), mysql_db_query(), mysql_drop_db(), mysql_errno(), mysql_error(), mysql_insert_id(), mysql_list_dbs(), mysql_list_fields(), mysql_list_tables(), mysql_query(), mysql_result(), mysql_select_db(), mysql_tablename(), mysql_get_host_info(), mysql_get_proto_info(), mysql_get_server_info()	Nadie	Enlace persistente con base de datos MySQL
mysql result	mysql_db_query(), mysql_list_dbs(), mysql_list_fields(), mysql_list_tables(), mysql_query()	mysql_data_seek(), mysql_db_name(), mysql_fetch_array(), mysql_fetch_assoc(), mysql_fetch_field(), mysql_fetch_lengths(), mysql_fetch_object(), mysql_fetch_row(), mysql_fetch_row(), mysql_field_flags(), mysql_field_name(), mysql_field_len(), mysql_field_seek(), mysql_field_table(), mysql_field_type(), mysql_num_fields(), mysql_num_rows(), mysql_result(), mysql_tablename()	mysql_free_result()	Resultado MySQL
oci8 collection				
oci8 connection	ocilogon(), ociplogon(), ocinlogon()	ocicommit(), ociserverversion(), ocinewcursor(), ociparse(), ocierror()	ocilogoff()	Enlace con base de datos Oracle
oci8 descriptor				
oci8 server				
oci8 session				

Nombre del Tipo de Recurso	Creado Por	Usado Por	Destruido Por	Definición
oci8 statement	ocinewdescriptor()	ocirollback(), ocinewdescriptor(), ocirowcount(), ocidefinebyname(), ocibindbyname(), ociexecute(), ocinumcols(), ociresult(), ocifetch(), ocifetchinto(), ocifetchstatement(), ocicolumnisnull(), ocicolumnname(), ocicolumnsize(), ocicolumntype(), ocistatementtype(), ocierror()	ocifreestatement()	Cursor Oracle
odbc link	odbc_connect()	odbc_autocommit(), odbc_commit(), odbc_error(), odbc_errormsg(), odbc_exec(), odbc_tables(), odbc_tableprivileges(), odbc_do(), odbc_prepare(), odbc_columns(), odbc_columnprivileges(), odbc_procedurecolumns(), odbc_specialcolumns(), odbc_rollback(), odbc_setoption(), odbc_gettypeinfo(), odbc_primarykeys(), odbc_foreignkeys(), odbc_procedures(), odbc_statistics()	odbc_close()	Enlace con base de datos ODBC

Nombre del Tipo de Recurso	Creado Por	Usado Por	Destruido Por	Definición
odbc link persistent	odbc_connect()	odbc_autocommit(), odbc_commit(), odbc_error(), odbc_errormsg(), odbc_exec(), odbc_tables(), odbc_tableprivileges(), odbc_do(), odbc_prepare(), odbc_columns(), odbc_columnprivileges(), odbc_procedurecolumns(), odbc_specialcolumns(), odbc_rollback(), odbc_setoption(), odbc_gettypeinfo(), odbc_primarykeys(), odbc_foreignkeys(), odbc_procedures(), odbc_statistics()	Nadie	Enlace persistente con base de datos ODBC
odbc result	odbc_prepare()	odbc_binmode(), odbc_cursor(), odbc_execute(), odbc_fetch_into(), odbc_fetch_row(), odbc_field_name(), odbc_field_num(), odbc_field_type(), odbc_field_len(), odbc_field_precision(), odbc_field_scale(), odbc_longreadlen(), odbc_num_fields(), odbc_num_rows(), odbc_result(), odbc_result_all(), odbc_setoption()	odbc_free_result()	Resultado ODBC
birdstep link				
birdstep result				
OpenSSL key	openssl_get_privatekey(), openssl_get_publickey()	openssl_sign(), openssl_seal(), openssl_open(), openssl_verify()	openssl_free_key()	Llave OpenSSL
OpenSSL X.509	openssl_x509_read()	openssl_x509_parse(), openssl_x509_checkpurpose()	openssl_x509_free()	Llave pública

Nombre del Tipo de Recurso	Creado Por	Usado Por	Destruido Por	Definición
oracle Cursor	**ora_open()**	**ora_bind()**, **ora_columnname()**, **ora_columnsize()**, **ora_columntype()**, **ora_error()**, **ora_errorcode()**, **ora_exec()**, **ora_fetch()**, **ora_fetch_into()**, **ora_getcolumn()**, **ora_numcols()**, **ora_numrows()**, **ora_parse()**	**ora_close()**	Cursor Oracle
oracle link	**ora_logon()**	**ora_do()**, **ora_error()**, **ora_errorcode()**, **ora_rollback()**, **ora_commitoff()**, **ora_commiton()**, **ora_open()**, **ora_commit()**	**ora_logoff()**	Enlace con base de datos oracle
oracle link persistent	**ora_plogon()**	**ora_do()**, **ora_error()**, **ora_errorcode()**, **ora_rollback()**, **ora_commitoff()**, **ora_commiton()**, **ora_open()**, **ora_commit()**	Nadie	Enlace persistente con base de datos oracle

Nombre del Tipo de Recurso	Creado Por	Usado Por	Destruido Por	Definición
pdf document	pdf_new()	pdf_add_bookmark(), pdf_add_launchlink(), pdf_add_locallink(), pdf_add_note(), pdf_add_pdflink(), pdf_add_weblink(), pdf_arc(), pdf_attach_file(), pdf_begin_page(), pdf_circle(), pdf_clip(), pdf_closepath(), pdf_closepath_fill_stroke(), pdf_closepath_stroke(), pdf_concat(), pdf_continue_text(), pdf_curveto(), pdf_end_page(), pdf_endpath(), pdf_fill(), pdf_fill_stroke(), pdf_findfont(), pdf_get_buffer(), pdf_get_image_height(), pdf_get_image_width(), pdf_get_parameter(), pdf_get_value(), pdf_lineto(), pdf_moveto(), pdf_open_ccitt(), pdf_open_file(), pdf_open_image_file(), pdf_place_image(), pdf_rect(), pdf_restore(), pdf_rotate(), pdf_save(), pdf_scale(), pdf_setdash(), pdf_setflat(), pdf_setfont(), pdf_setgray(), pdf_setgray_fill(), pdf_setgray_stroke(), pdf_setlinecap(), pdf_setlinejoin(), pdf_setlinewidth(), pdf_setmiterlimit(), pdf_setpolydash(), pdf_setrgbcolor(), pdf_setrgbcolor_fill(), pdf_setrgbcolor_stroke(), pdf_set_border_color(), pdf_set_border_dash(), pdf_set_border_style(), pdf_set_char_spacing(), pdf_set_duration(),	pdf_close(), pdf_delete()	Documento PDF

Nombre del Tipo de Recurso	Creado Por	Usado Por	Destruido Por	Definición
pdf image	pdf_open_image(), pdf_open_image_file(), pdf_open_memory_image()	pdf_get_image_height(), pdf_get_image_width(), pdf_open_CCITT(), pdf_place_image()	pdf_close_image()	Imagen en archivo PDF
pdf object				
pdf outline				
pgsql large object	pg_lo_open()	pg_lo_open(), pg_lo_create(), pg_lo_read(), pg_lo_read_all(), pg_lo_seek(), pg_lo_tell(), pg_lo_unlink(), pg_lo_write()	pg_lo_close()	Objato Largo PostgreSQL
pgsql link	pg_connect()	pg_affected_rows(), pg_query(), pg_send_query(), pg_get_result(), pg_connection_busy(), pg_connection_reset(), pg_connection_status(), pg_last_error(), pg_last_notice(), pg_lo_create(), pg_lo_export(), pg_lo_import(), pg_lo_open(), pg_lo_unlink(), pg_host(), pg_port(), pg_dbname(), pg_options(), pg_copy_from(), pg_copy_to(), pg_end_copy(), pg_put_line(), pg_tty(), pg_trace(), pg_untrace(), pg_set_client_encoding(), pg_client_encoding(), **pg_metadata()**, pg_convert(), pg_insert(), pg_select(), pg_delete(), pg_update()	pg_close()	Enlace con base de datos PostgreSQL

Nombre del Tipo de Recurso	Creado Por	Usado Por	Destruido Por	Definición
pgsql link persistent	pg_pconnect()	pg_affected_rows(), pg_query(), pg_send_query(), pg_get_result(), pg_connection_busy(), pg_connection_reset(), pg_connection_status(), pg_last_error(), pg_last_notice(), pg_lo_create(), pg_lo_export(), pg_lo_import(), pg_lo_open(), pg_lo_unlink(), pg_host(), pg_port(), pg_dbname(), pg_options(), pg_copy_from(), pg_copy_to(), pg_end_copy(), pg_put_line(), pg_tty(), pg_trace(), pg_untrace(), pg_set_client_encoding(), pg_client_encoding(), **pg_metadata()**, pg_convert(), pg_insert(), pg_select(), pg_delete(), pg_update()	Nadie	Enlace persistente con base de datos PostgreSQL
pgsql result	pg_query(), pg_get_result()	pg_fetch_array(), pg_fetch_object(), pg_fetch_result(), pg_fetch_row(), pg_field_is_null(), pg_field_name(), pg_field_num(), pg_field_prtlen(), pg_field_size(), pg_field_type(), pg_last_oid(), pg_num_fields(), pg_num_rows(), pg_result_error(), pg_result_status()	pg_free_result()	Resultado PostgreSQL
pgsql string				
printer				
printer brush				
printer font				
printer pen				

Nombre del Tipo de Recurso	Creado Por	Usado Por	Destruido Por	Definición
pspell	pspell_new(), pspell_new_config(), pspell_new_personal()	pspell_add_to_personal(), pspell_add_to_session(), pspell_check(), pspell_clear_session(), pspell_config_ignore(), pspell_config_mode(), pspell_config_personal(), pspell_config_repl(), pspell_config_runtogether(), pspell_config_save_repl(), pspell_save_wordlist(), pspell_store_replacement(), pspell_suggest()	Nadie	Diccionario pspell
pspell config	pspell_config_create()	pspell_new_config()	Nadie	Configuración pspell
Sablotron XSLT	xslt_create()	**xslt_closelog()**, **xslt_openlog()**, **xslt_run()**, xslt_set_sax_handler(), xslt_errno(), xslt_error(), **xslt_fetch_result()**, xslt_free()	xslt_free()	Analizador sintáctico XSLT
shmop	shmop_open()	shmop_read(), shmop_write(), shmop_size(), shmop_delete()	shmop_close()	
sockets file descriptor set	**socket()**	**accept_connect()**, **bind()**, **connect()**, **listen()**, **read()**, **write()**	**close()**	Socket
sockets i/o vector				
dir	**dir()**	readdir(), rewinddir()	closedir()	Gestor de directorio
stream	fopen()	feof(), fflush(), fgetc(), fgetcsv(), fgets(), fgetss(), flock(), fpassthru(), fputs(), fwrite(), fread(), fseek(), ftell(), fstat(), ftruncate(), set_file_buffer(), rewind()	fclose()	Gestor de archivo
pipe	popen()	feof(), fflush(), fgetc(), fgetcsv(), fgets(), fgetss(), fpassthru(), fputs(), fwrite(), fread()	pclose()	Gestor de proceso

Nombre del Tipo de Recurso	Creado Por	Usado Por	Destruido Por	Definición
socket	fsockopen()	fflush(), fgetc(), fgetcsv(), fgets(), fgetss(), fpassthru(), fputs(), fwrite(), fread()	fclose()	Gestor de socket
sybase-db link	sybase_connect()	sybase_query(), sybase_select_db()	sybase_close()	Enlace a base de datos Sybase usando la biblioteca DB
sybase-db link persistent	sybase_pconnect()	sybase_query(), sybase_select_db()	Nadie	Enlace persistente con base de datos Sybase usando la biblioteca DB
sybase-db result	sybase_query()	sybase_data_seek(), sybase_fetch_array(), sybase_fetch_field(), sybase_fetch_object(), sybase_fetch_row(), sybase_field_seek(), sybase_num_fields(), sybase_num_rows(), sybase_result()	sybase_free_result()	Resultado Sybase usando la biblioteca DB
sybase-ct link	sybase_connect()	sybase_affected_rows(), sybase_query(), sybase_select_db()	sybase_close()	Enlace a base de datos Sybase usando la biblioteca CT
sybase-ct link persistent	sybase_pconnect()	sybase_affected_rows(), sybase_query(), sybase_select_db()	Nadie	Enlace persistente con base de datos Sybase usando la biblioteca CT
sybase-ct result	sybase_query()	sybase_data_seek(), sybase_fetch_array(), sybase_fetch_field(), sybase_fetch_object(), sybase_fetch_row(), sybase_field_seek(), sybase_num_fields(), sybase_num_rows(), sybase_result()	sybase_free_result()	Resultado Sybase usando la biblioteca CT
sysvsem	sem_get()	sem_acquire()	sem_release()	Semáforo de Sistema V

Nombre del Tipo de Recurso	Creado Por	Usado Por	Destruido Por	Definición
sysvshm	shm_attach()	shm_remove(), shm_put_var(), shm_get_var(), shm_remove_var()	shm_detach()	Memoria Compartida de Sistema V
wddx	wddx_packet_start()	wddx_add_vars()	wddx_packet_end()	Paquete WDDX
xml	xml_parser_create()	xml_set_object(), xml_set_element_handler(), xml_set_character_data_handler(), xml_set_processing_instruction_handler(), xml_set_default_handler(), xml_set_unparsed_entity_decl_handler(), xml_set_notation_decl_handler(), xml_set_external_entity_ref_handler(), xml_parse(), xml_get_error_code(), xml_error_string(), xml_get_current_line_number(), xml_get_current_column_number(), xml_get_current_byte_index(), xml_parse_into_struct(), xml_parser_set_option(), xml_parser_get_option()	xml_parser_free()	Analizador sintáctico XML
zlib	gzopen()	gzeof(), gzgetc(), gzgets(), gzgetss(), gzpassthru(), gzputs(), gzread(), gzrewind(), gzseek(), gztell(), gzwrite()	gzclose()	Archivo comprimido gz

Apéndice L. Lista de Protocolos/Envolturas Soportadas

La siguiente es una lista de los varios protocolos estilo URL que PHP tiene integrado para su uso con las funciones del sistema de archivos, tales como fopen() y copy(). Adicionalmente a estas envolturas, y a partir de PHP 4.3.0, usted puede escribir sus propias envolturas usando scripts PHP y stream_wrapper_register().

Sistema de archivos

Todas las versiones de PHP. Usada explícitamente mediante `file://` a partir de PHP 4.3.0

- `/ruta/hacia/archivo.ext`
- `ruta/relativa/hacia/archivo.ext`
- `archivoEnDirActual.ext`
- `C:/ruta/hacia/archivo_win.ext`
- `C:\ruta\hacia\archivo_win.ext`
- `\\servidor_smb\recurso_compartido\ruta\hacia\archivo_win.ext`
- `file:///ruta/hacia/archivo.ext`

`file://` es la envoltura predeterminada usada por PHP, y representa el sistema de archivos local. Cuando se especifica una ruta relativa (una ruta que no comienza con /, \, \\, o una letra de unidad en windows), la ruta provista será aplicada contra el directorio de trabajo actual. En muchos casos éste es el directorio en el cual reside el script, a menos que haya sido modificado. Al usar la sapi CLI, éste es, por defecto, el directorio desde donde fue llamado el script.

Con algunas funciones, como **fopen()** y **file_get_contents()**, *include_path* puede usarse opcionalmente también para las búsquedas de rutas relativas.

Tabla L-1. Resumen de Envoltura

Atributo	Soporte
Restricción por allow_url_fopen.	No
Permite Lectura	Si
Permite Escritura	Si
Permite Adición	Si
Permite Lectura y Escritura Simultánea	Si
Soporte **stat()**	Si
Soporte **unlink()**	Si
Soporte **rename()**	Si
Soporte **mkdir()**	Si
Soporte **rmdir()**	Si

HTTP y HTTPS

PHP 3, PHP 4, PHP 5. `https://` a partir de PHP 4.3.0

- `http://example.com`
- `http://example.com/archivo.php?var1=val1&var2=val2`

- `http://usuario:contrasenya@example.com`
- `https://example.com`
- `https://example.com/archivo.php?var1=val1&var2=val2`
- `https://usuario:contrasenya@example.com`

Permite acceso de sólo-lectura a archivos/recursos a través de HTTP 1.0, usando el método HTTP GET. Una cabecera *Host:* es enviada con la petición para gestionar hosts virtuales basados en nombres. Si ha configurado una cadena user_agent usando su archivo ini o en el contexto de secuencia, también ésta será usada en la petición.

Aviso
Cuando se usa SSL, Microsoft IIS violara el protocolo, cerrando la conexion sin mandar un indicador close_notify. PHP avisara de esto con este mensaje "SSL: Fatal Protocol Error", cuando llegue al final de los datos. Una solucion a este problema es bajar el nivel de aviso de errores del sistema para que no incluya advertencias. PHP 4.3.7 y versiones posteriores detectan servidores IIS con este problema y suprime la advertencia. Si usais la funcion **fsockopen()** para crear un socket ssl://, tendreis que suprimir la advertencia explicitamente.

Las redirecciones han sido soportadas desde PHP 4.0.5; si se encuentra usando una versión anterior, necesitará incluir barras de cierre en sus URLs. Si es importante conocer la URL del recurso del cual proviene su documento (luego de que todas las redirecciones han sido procesadas), necesitará trabajar con la serie de cabeceras de respuesta devueltas por la secuencia.

```
<?php
$url = 'http://www.example.com/pagina_de_redireccion.php';

$da = fopen($url, 'r');

/* Antes de PHP 4.3.0, use $http_response_header
   en lugar de stream_get_meta_data() */
foreach(stream_get_meta_data($da) as $respuesta) {

  /* Fuimos redireccionados? */
  if (substr(strtolower($respuesta), 0, 10) == 'location: ') {
    /* actualizar $url con la ubicacion desde donde fuimos
       redireccionados */
    $url = substr($respuesta, 10);
  }

}

?>
```

La secuencia permite acceso al *cuerpo* del recurso; las cabeceras son almacenadas en la variable *$http_response_header*. A partir de PHP 4.3.0, las cabeceras están disponibles mediante el uso de **stream_get_meta_data()**.

Las conexiones HTTP son de sólo-lectura; no puede escribir datos o copiar archivos hacia un recurso HTTP.

> **Nota:** HTTPS es soportado a partir de PHP 4.3.0, si ha compilado el soporte para OpenSSL.

Tabla L-2. Resumen de Envoltura

Atributo	Soporte
Restricción por allow_url_fopen.	Si
Permite Lectura	Si
Permite Escritura	No
Permite Adición	No
Permite Lectura y Escritura Simultánea	N/D
Soporte **stat()**	No
Soporte **unlink()**	No
Soporte **rename()**	No
Soporte **mkdir()**	No
Soporte **rmdir()**	No

Tabla L-3. Opciones de contexto (a la altura de PHP 5.0.0)

Nombre	Uso	Predeterminado
method	**GET**, **POST**, u otro método HTTP soportado por el servidor remoto.	**GET**
header	Cabeceras adicionales a ser enviadas durante la petición. Los valores en esta opción sobrescribirán otros valores (tales como *User-agent:*, *Host:*, y *Authentication:*).	
user_agent	Valor a enviar con la cabecera User-Agent:. Este valor solo será usado si user-agent *no* es especificado en la opción de contexto *header* anterior.	Parámetro `php.ini`: *user_agent*
content	Datos adicionales a ser enviados después de las cabeceras. Típicamente usados con peticiones POST o PUT.	
proxy	URI que especifica la dirección del servidor proxy. (p.ej. *tcp://proxy.example.com:5100*).	
request_fulluri	Cuando es definido como **TRUE**, la URI completa será usada a la hora de construir la petición. (Es decir, *GET http://www.example.com/ruta/hacia/archivo.html HTTP/1.0*). Aunque éste es un formato de petición no-estándar, algunos servidores proxy lo requieren.	**FALSE**

Opciones de contexto de las secuencias de socket interno: Puede que se soporten opciones de contexto adicionales por el transporte implícito. Para secuencias *http://*, refiérase a las opciones de contexto para el transporte *tcp://*. Para secuencias *https://*, refiérase a las opciones de contexto del transporte *ssl://*.

FTP y FTPS

PHP 3, PHP 4, PHP 5. `ftps://` a partir de PHP 4.3.0

- `ftp://example.com/pub/archivo.txt`
- `ftp://usuario:contrasenya@example.com/pub/archivo.txt`

- `ftps://example.com/pub/archivo.txt`
- `ftps://usuario:contrasenya@example.com/pub/archivo.txt`

Permite acceso de lectura a archivos exsistentes y la creación de nuevos archivos mediante FTP. Si el servidor no soporta el modo pasivo de ftp, la conexión fallará.

Puede abrir archivos para lectura o para escritura, pero no en los dos modos simultáneamente. Si el archivo remoto existe previamente en el servidor ftp e intenta abrirlo para escritura, pero no tiene especificada la opción de contexto *overwrite*, la conexión fallará. Si necesita sobrescribir archivos existentes sobre ftp, especifique la opción *overwrite* en el contexto y abra el archivo para escritura. Alternativamente, puede usar la extensión FTP.

> **Adición:** A partir de PHP 5.0.0, los archivos pueden ser extendidos, agregando nuevos datos al final, mediante la envoltura de URL *ftp://*. En versiones anteriores, al intentar adicionar datos a un archivo mediante *ftp://* se producirá un fallo.

`ftps://` se ha introducido en PHP 4.3.0. Es igual a `ftp://`, pero intenta negociar una conexión segura con el servidor ftp. Si el servidor no soporta SSL, entonces la conexión recae sobre el ftp corriente, sin encripción.

> **Nota:** FTPS es soportado desde PHP 4.3.0, si ha compilado el soporte para OpenSSL.

Tabla L-4. Resumen de Envoltura

Atributo	PHP 4	PHP 5
Restricción por allow_url_fopen.	Si	Si
Permite Lectura	Si	Si
Permite Escritura	Si (archivos nuevos únicamente)	Si (archivos nuevos/archivos existentes con *overwrite*)
Permite Adición	No	Si
Permite Lectura y Escritura Simultánea	No	No
Soporte **stat()**	No	A partir de PHP 5.0.0: Sólo los elementos **filesize()**, **filetype()**, **file_exists()**, **is_file()**, e **is_dir()**. A partir de PHP 5.1.0: **filemtime()**.
Soporte **unlink()**	No	Si
Soporte **rename()**	No	Si
Soporte **mkdir()**	No	Si
Soporte **rmdir()**	No	Si

Tabla L-5. Opciones de contexto (a la altura de PHP 5.0.0)

Nombre	Uso	Predeterminado
overwrite	Permite la sobrescritura de archivos previamente existentes en el servidor remoto. Se aplica al modo de escritura (carga de archivos) solamente.	**FALSE** (Deshabilitada)

Nombre	Uso	Predeterminado
resume_pos	Desplazamiento de archivo para iniciar la transferencia. Se aplica únicamente al modo de lectura (descarga de archivos).	0 (Comienzo del Archivo)
proxy(PHP 5.1.0 o superior)	Petición proxy FTP mediante un servidor proxy http. Se aplica únicamente a las operaciones de lectura de archivos. Ej: *tcp://squid.example.com:8000*	

Opciones de contexto de las secuencias de socket interno: Puede que se soporten opciones de contexto adicionales por el transporte implícito. Para secuencias *ftp://*, refiérase a las opciones de contexto para el transporte *tcp://*. Para secuencias *ftps://*, refiérase a las opciones de contexto del transporte *ssl://*.

Secuencias de entrada/salida PHP

PHP 3.0.13 y superior, `php://output` y `php://input` a partir de PHP 4.3.0, `php://filter` desde PHP 5.0.0

- `php://stdin`
- `php://stdout`
- `php://stderr`
- `php://output`
- `php://input`
- `php://filter`

`php://stdin`, `php://stdout` y `php://stderr` le ofrecen acceso a las secuencias de entrada o salida correspondientes del proceso PHP.

`php://output` le permite escribir sobre el mecanismo de búfer de salida en la misma manera que lo hacen **print()** y **echo()**.

`php://input` le permite leer datos POST en su forma primitiva. Es una alternativa que consume menos memoria que *$HTTP_RAW_POST_DATA* y no requiere de directivas `php.ini` especiales.

`php://stdin` y `php://input` son de sólo-lectura, mientras que `php://stdout`, `php://stderr` y `php://output` son de sólo-escritura.

`php://filter` es una especie de meta-envoltura diseñada para permitir el uso de filtros sobre una secuencia al momento de su apertura. Resulta útil con aquellas funciones de archivos todo-en-uno, como **readfile()**, **file()**, y **file_get_contents()** en donde de otro modo no habría forma de aplicar filtros a la secuencia antes de que los contenidos fueran leídos.

El destino `php://filter` recibe los siguientes 'parámetros' como partes de su 'ruta'.

- */resource=<secuencia a ser filtrada>* (*requerido*) Este parámetro debe estar ubicado el final de su especificación `php://filter` y debe apuntar a la secuencia que desea filtrar.

```
<?php
/* Esto es ñalente a simplemente:
   readfile("http://www.example.com");
   ya que no se aplica filtro alguno */

readfile("php://filter/resource=http://www.example.com");
?>
```

- */read=<lista de filtros a aplicar a la cadena de lectura>* (*opcional*) Este parámetro toma uno o más nombres de filtros separados por el caracter |.

```
<?php
/* Esto produce como salida el contenido de
   www.example.com enteramente en mayusculas */
readfile("php://filter/read=string.toupper/resource=http://www.example.com");

/* Esto hace lo mismo pero tambien codifica la salida con  ROT13 */
readfile("php://filter/read=string.toupper|string.rot13/resource=http://www.examp
?>
```

- */write=<lista de filtros a aplicar a la cadena de escritura>* (*opcional*) Este parámetro toma uno o más nombres de filtro separados por el caracter |.

```
<?php
/* Esto filtra la cadena "Hola Mundo" a traves del filtro rot13,
   luego escribe al archivo ejemplo.txt en el directorio actual */
file_put_contents("php://filter/write=string.rot13/resource=ejemplo.txt","Hola Mu
?>
```

- */<lista de filtros a aplicar a ambas cadenas>* (*opcional*) Cualquier lista de filtros que no esté precedida por *read=* o *write=* será aplicada tanto a la cadena de lectura como a la de escritura (según sea el caso).

Tabla L-6. Resumen de Envoltura (Para *php://filter*, refiérase al resumen de la envoltura siendo filtrada.)

Atributo	Soporte
Restricción por allow_url_fopen.	No
Permite Lectura	*php://stdin* y *php://input* únicamente.
Permite Escritura	*php://stdout*, *php://stderr*, y *php://output* únicamente.
Permite Adición	*php://stdout*, *php://stderr*, y *php://output* únicamente. (ñalente a la escritura)
Permite Lectura y Escritura Simultánea	No. Estas envolturas son unidireccionales.
Soporte **stat()**	No
Soporte **unlink()**	No
Soporte **rename()**	No
Soporte **mkdir()**	No
Soporte **rmdir()**	No

Secuencias de Compresión

`zlib:` PHP 4.0.4 - PHP 4.2.3 (sólo sistemas con fopencookie)

`compress.zlib://` y `compress.bzip2://` PHP 4.3.0 y superior

- `zlib:`
- `compress.zlib://`
- `compress.bzip2://`

`zlib:` funciona como **gzopen()**, con la excepción de que la secuencia puede ser usada con **fread()** y otras funciones del sistema de archivos. Esto se ha hecho obsoleto en PHP 4.3.0 debido a las ambigüedades presentes con nombres de archivos que contienen caracteres ':'; use `compress.zlib://` en su lugar.

`compress.zlib://` y `compress.bzip2://` son ñalentes a **gzopen()** y **bzopen()** respectivamente, y operan incluso sobre sistemas que no tienen soporte para fopencookie.

Tabla L-7. Resumen de Envoltura

Atributo	Soporte
Restricción por allow_url_fopen.	No
Permite Lectura	Si
Permite Escritura	Si
Permite Adición	Si
Permite Lectura y Escritura Simultánea	No
Soporte **stat()**	No, use la envoltura *file://* normal para realizar stat sobre archivos comprimidos.
Soporte **unlink()**	No, use la envoltura *file://* normal para eliminar archivos comprimidos.
Soporte **rename()**	No
Soporte **mkdir()**	No
Soporte **rmdir()**	No

Shell Seguro 2

`ssh2.shell://` `ssh2.exec://` `ssh2.tunnel://` `ssh2.sftp://` `ssh2.scp://` PHP 4.3.0 y superiores (PECL)

- `ssh2.shell://usuario:contrasenya@example.com:22/xterm`
- `ssh2.exec://usuario:contrasenya@example.com:22/usr/local/bin/algun_comando`
- `ssh2.tunnel://usuario:contrasenya@example.com:22/192.168.0.1:14`
- `ssh2.sftp://usuario:contrasenya@example.com:22/ruta/hacia/archivo`

Esta envoltura no se encuentra habilitada por defecto: Para usar las envolturas `ssh2.*://`, es necesario instalar la extensión SSH2 disponible desde PECL.

Además de aceptar detalles de inicio de sesión tradicionales tipo URI, las envolturas ssh2 reusan conexiones abiertas pasando el recurso de conexión en la porción host de la URL.

Ejemplo L-1. Abrir una secuencia desde una conexión activa

```
<?php
$sesion = ssh2_connect('example.com', 22);
ssh2_auth_pubkey_file($sesion, 'nombre_usuario',
                      '/home/usuario/.ssh/id_rsa.pub',
                      '/home/usuario/.ssh/id_rsa', 'secreto');
$secuencia = fopen("ssh2.tunnel://$sesion/remote.example.com:1234", 'r');
?>
```

Tabla L-8. Resumen de Envoltura

Atributo	ssh2.shell	ssh2.exec	ssh2.tunnel	ssh2.sftp	ssh2.scp
Restringido por allow_url_fopen.	Si	Si	Si	Si	Si
Permite lectura	Si	Si	Si	Si	Si
Permite escritura	Si	Si	Si	Si	No
Permite adición	No	No	No	Si (Cuando lo permite el servidor)	No
Permite Lectura y Escritura Simultánea	Si	Si	Si	Si	No
Soporta **stat()**	No	No	No	Si	No
Soporta **unlink()**	No	No	No	Si	No
Soporta **rename()**	No	No	No	Si	No
Soporta **mkdir()**	No	No	No	Si	No
Soporta **rmdir()**	No	No	No	Si	No

Tabla L-9. Opciones de contexto

Nombre	Uso	Predeterminado
session	Recurso ssh2 preconectado para reusar	
sftp	Recurso sftp prereservado para reusar	
methods	Métodos de intercambio de llave, llave de host, cifrado, compresión y MAC a usar	
callbacks		
username	Nombre de usuario para la conexión	
password	Contraseña a usar con autenticación de contraseñas	
pubkey_file	Nombre del archivo de llave pública a usar para la autenticación	
privkey_file	Nombre del archivo de llave privada a usar para la autenticación	
env	Matriz asociativa de variables de entorno a definir	

Nombre	Uso	Predeterminado
term	Tipo de emulación de terminal para solicitar cuando se reserva una pty	
term_width	Ancho de la terminal solicitada cuando se reserva una pty	
term_height	Altura de la terminal solicitada cuando se reserva una pty	
term_units	Unidades a usar con term_width y term_height	**SSH2_TERM_UNIT_CHARS**

Secuencias de Audio

`ogg://` PHP 4.3.0 y superior (PECL)

- `ogg://archivo_sonido.ogg`
- `ogg:///ruta/hacia/archivo_sonido.ogg`
- `ogg://http://www.example.com/ruta/hacia/secuencia_sonido.ogg`

Esta envoltura no está habilitada de manera predeterminada: Para poder usar la envoltura `ogg://`, necesita instalar la extensión OGG/Vorbis, disponible desde PECL.

Los archivos abiertos para lectura mediante la envoltura `ogg://` son tratados como audio codificado abierto usando el codec OGG/Vorbis. De forma semejante, los archivos abiertos para escritura o adición a través de la envoltura `ogg://` son escritos como datos de audio comprimido. Cuando se usa **stream_get_meta_data()** sobre un archivo OGG/Vorbis abierto para lectura, devolverá varios detalles sobre la secuencia, incluyendo la etiqueta *vendor*, cualquier comentario incluido en *comments*, el número de canales en *channels*, la tasa de muestra en *rate*, y el rango de codificación descrito por: *bitrate_lower*, *bitrate_upper*, *bitrate_nominal*, y *bitrate_window*.

Tabla L-10. Resumen de Envoltura

Atributo	Soporte
Restricción por allow_url_fopen.	No
Permite Lectura	Si
Permite Escritura	Si
Permite Adición	Si
Permite Lectura y Escritura Simultánea	No
Soporte **stat()**	No
Soporte **unlink()**	No
Soporte **rename()**	No
Soporte **mkdir()**	No
Soporte **rmdir()**	No

Tabla L-11. Opciones de contexto

Nombre	Uso	Predeterminado	Modo
pcm_mode	Codificación PCM a aplicar durante la lectura, un valor entre: **OGGVORBIS_PCM_U8**, **OGGVORBIS_PCM_S8**, **OGGVORBIS_PCM_U16_BE**, **OGGVORBIS_PCM_S16_BE**, **OGGVORBIS_PCM_U16_LE**, y **OGGVORBIS_PCM_S16_LE**. (8 vs 16 bit, con signo o sin signo, big-endian o little-endian)	OGGVORBIS_PCM_S16_LE	Lectura
rate	Tasa de muestra de datos de entrada, expresada en Hz	44100	Escritura/ Adición
bitrate	Cuando es un valor entero, la tasa de bits fija para codificar. (16000 a 131072) Cuando se define como flotante, la calidad de tasa de bits variable a usar. (-1.0 a 1.0)	128000	Escritura/ Adición
channels	El número de canales de audio a codificar, normalmente 1 (Mono), o 2 (Stero). Su rango puede llegar hasta 16.	2	Escritura/ Adición
comments	Una matriz de valores de cadena para codificar en la cabecera del material.		Escritura/ Adición

---

Apéndice M. Lista de Filtros Disponibles

La siguiente es una lista de unos cuantos filtros de secuencia integrados que puede usar con **stream_filter_append()**. Su versión de PHP puede tener más (o menos) filtros de los que son listados aquí.

Vale la pena anotar una ligera asimetría entre **stream_filter_append()** y **stream_filter_prepend()**. Cada secuencia en PHP contiene un pequeño *búfer de lectura* en donde almacena los bloques de datos recibidos desde el sistema de archivos u otro recurso para procesar los datos de la forma más eficiente. Tan pronto como los datos son tomados del recurso y colocados en el búfer interno de la secuencia, éstos son procesados inmediatamente a través de cualquier filtro adjunto sin importar que la aplicación PHP esté lista para los datos o no. Si los datos están en el búfer de lectura cuando un filtro es *añadido al final*, los datos serán procesados inmediatamente a través del filtro haciendo parecer que el proceso fuera transparente. Sin embargo, si los datos están en el búfer de lectura cuando un filtro es *adjuntado al comienzo*, los datos *NO* serán procesados a través del filtro. En su lugar esperará a que el siguiente bloque de datos sea recuperado del recurso.

Para una lista de filtros instalados en su versión de PHP, use **stream_get_filters()**.

Filtros de Cadena

Cada uno de estos filtros hace precisamente lo que sus nombres implican y corresponden al comportamiento de una función de manipulación de cadenas que hace parte de php. Para más información sobre un filtro dado, refiérase a la página del manual de la función correspondiente.

string.rot13 (a partir de PHP 4.3.0) El uso de este filtro es ñalente a procesar todos los datos de la secuencia a través de la función **str_rot13()**.

Ejemplo M-1. string.rot13

```
<?php
$da = fopen('php://output', 'w');
stream_filter_append($da, 'string.rot13');
fwrite($da, "Esto es una prueba.\n");
/* Imprime:  Rfgb rf han cehron. */
?>
```

string.toupper (a partir de PHP 5.0.0) El uso de este filtro es ñalente a procesar todos los datos de la secuencia a través de la función **strtoupper()**.

Ejemplo M-2. string.toupper

```
<?php
$da = fopen('php://output', 'w');
stream_filter_append($da, 'string.toupper');
fwrite($da, "Esto es una prueba.\n");
/* Imprime:  ESTO ES UNA PRUEBA.   */
?>
```

string.tolower (a partir de PHP 5.0.0) El uso de este filtro es ñalente a procesar todos los datos de la secuencia a través de la función **strtolower()**.

Ejemplo M-3. string.tolower

```
<?php
$da = fopen('php://output', 'w');
stream_filter_append($da, 'string.tolower');
fwrite($da, "Esto es una prueba.\n");
/* Imprime:  esto es una prueba.   */
?>
```

string.strip_tags (a partir de PHP 5.0.0) El uso de este filtro es ñalente a procesar todos los datos de la secuencia a través de la función **strip_tags()**. Acepta parámetros en una de dos formas: Ya sea como una cadena que contiene una lista de etiquetas, similar al segundo parámetro de la función **strip_tags()**, o como una matriz de nombres de etiqueta.

Ejemplo M-4. string.strip_tags

```
<?php
$da = fopen('php://output', 'w');
stream_filter_append($da, 'string.strip_tags', STREAM_FILTER_WRITE, "<b><i><u>");
fwrite($da, "<b>texto en negrilla</b> llevado a <h1>nivel de cabecera 1</h1>\n");
fclose($da);
/* Imprime:  <b>texto en negrilla</b> llevado a nivel de cabecera 1   */

$da = fopen('php://output', 'w');
stream_filter_append($da, 'string.strip_tags', STREAM_FILTER_WRITE, array('b','i','u'));
fwrite($da, "<b>texto en negrilla</b> llevado a <h1>nivel de cabecera 1</h1>\n");
fclose($da);
/* Imprime:  <b>texto en negrilla</b> llevado a nivel de cabecera 1   */
?>
```

Filtros de Conversión

De forma similar a los filtros string.*, los filtros convert.* realizan acciones semejantes a sus nombres. Los filtros de conversión fueron agregados en PHP 5.0.0. Para más información sobre un filtro dado, refiérase a la página del manual sobre la función correspondiente.

convert.base64-encode y *convert.base64-decode* El uso de estos filtros es ñalente a procesar todos los datos de la secuencia a través de las funciones **base64_encode()** y **base64_decode()** respectivamente. *convert.base64-encode* soporta el uso de parámetros entregados como una matriz asociativa. Si se usa *longitud-linea*, la salida base64 será separada en paquetes de *longitud-linea* caracteres cada uno. Si se usa *caracteres-final-de-linea*, cada paquete será delimitado por los

caracteres dados. Estos parámetros ofrecen el mismo efecto que usar **base64_encode()** con **chunk_split()**.

Ejemplo M-5. convert.base64-encode y convert.base64-decode

```
<?php
$da = fopen('php://output', 'w');
stream_filter_append($da, 'convert.base64-encode');
fwrite($da, "This is a test.\n");
fclose($da);
/* Imprime:  VGhpcyBpcyBhIHRlc3QuCg==  */

$param = array('line-length' => 8, 'line-break-chars' => "\r\n");
$da = fopen('php://output', 'w');
stream_filter_append($da, 'convert.base64-encode', STREAM_FILTER_WRITE, $param);
fwrite($da, "This is a test.\n");
fclose($da);
/* Imprime:  VGhpcyBp
          :  cyBhIHRl
          :  c3QuCg==  */

$da = fopen('php://output', 'w');
stream_filter_append($da, 'convert.base64-decode');
fwrite($da, "VGhpcyBpcyBhIHRlc3QuCg==");
fclose($da);
/* Imprime:  This is a test.  */
?>
```

convert.quoted-printable-encode y *convert.quoted-printable-decode* El uso de la versión de docedificación de este filtro es alente a procesar todos los datos de la secuencia a través de la función **quoted_printable_decode()**. No existe una función equivalente a *convert.quoted-printable-encode*. *convert.quoted-printable-encode* soporta el uso de parámetros dados como una matriz asociativa. Adicionalmente a los parámetros soportados por *convert.base64-encode*, *convert.quoted-printable-encode* soporta también los argumentos booleanos *binary* y *force-encode-first*. *convert.base64-decode* soporta únicamente el parámetro *line-break-chars* como una sugerencia del tipo para eliminar del material codificado.

Ejemplo M-6. convert.quoted-printable-encode y convert.quoted-printable-decode

```
<?php
$da = fopen('php://output', 'w');
stream_filter_append($da, 'convert.quoted-printable-encode');
fwrite($da, "Esto es una prueba.\n");
/* Imprime:  =Esto es una prueba.=0A  */
?>
```

Filtros de Compresión

Mientras que la sección de nombre *Secuencias de Compresión* en Apéndice L ofrece una forma de crear archivos compatibles con gzip y bz2 en el sistema de archivos local, no ofrece un método de compresión generalizado sobre secuencias de red, ni ofrece la forma de comenzar con una secuencia no-comprimida y trasladarse a una comprimida. Para esto, un filtro de compresión puede ser aplicado sobre cualquier recurso de secuencia en cualquier momento.

> **Nota:** Los filtros de compresión *no* generan cabeceras ni caracteres finales usados por utilidades de la línea de comandos como *gzip*. Sólo comprimen y descomprimen las porciones significativas de las secuencias de datos comprimidas.

zlib.deflate (compresión) y *zlib.inflate* (descompresión) son implementaciones de los métodos de compresión descritos en RFC 1951. El filtro *deflate* recibe hasta tres parámetros pasados como una matriz asociativa. *level* describe la intensidad de compresión (1-9). Números más grandes producen,

por lo general, resultados más pequeños, al costo de tiempo de procesamiento adicional. Dos niveles de compresión especiales existen también: 0 (para no-compresión), y -1 (el valor predeterminado de zlib -- actualmente 6). *window* es el registro en base-2 del tamaño de ventana del circuito cerrado de compresión. Valores más altos (de hasta 15 -- 32768 bytes) producen mejor compresión al costo de más memoria, mientras valores más bajos (alrededor de 9 -- 512 bytes) producen una compresión inferior en un espacio de memoria más pequeño. El tamaño de *window* es actualmente **15**. *memory* es una escala que indica cuánta memoria debería ser reservada. Los valores válidos están en el rango de 1 (reserva mínima) a 9 (reserva máxima). Esta reserva de memoria afecta únicamente la repidez y no tiene impacto en el tamaño del resultado generado.

> **Nota:** Dado que el nivel de compresión es el parámetro usado con mayor frecuencia, puede indicarse alternativamente como un valor entero simple (en lugar de un elemento tipo matriz).

Los filtros de compresión zlib.* están disponibles con PHP desde la versión *5.1.0* si el soporte Referencia CXLI, *Funciones de Compresión Zlib* se encuentra habilitado. También están disponibles como una característica portada de vuelta en la versión *5.0.x*, instalando el paquete zlib_filter desde PECL. Estos filtros *no* están disponibles para PHP 4.

Ejemplo M-7. *zlib.deflate* y *zlib.inflate*

```
<?php
$params = array('level' => 6, 'window' => 15, 'memory' => 9);

$texto_original = "This is a test.\nThis is only a test.\nThis is not an important stri
echo "El texto original tiene " . strlen($texto_original) . " caracteres.\n";

$da = fopen('test.deflated', 'w');
stream_filter_append($da, 'zlib.deflate', STREAM_FILTER_WRITE, $params);
fwrite($da, $texto_original);
fclose($da);

echo "El archivo comprimido tiene " . filesize('test.deflated') . " bytes.\n";
echo "El texto original era:\n";
/* Use readfile y zlib.inflate para descomprimir al vuelo */
readfile('php://filter/zlib.inflate/resource=test.deflated');

/* Genera la salida:

El texto original tiene 70 caracteres.
El archivo comprimido tiene 56 bytes.
El texto original era:
This is a test.
This is only a test.
This is not an important string.

 */
?>
```

Ejemplo M-8. *zlib.deflate* simple

```
<?php
$texto_original = "This is a test.\nThis is only a test.\nThis is not an important stri
echo "El texto original tiene " . strlen($texto_original) . " caracteres.\n";

$da = fopen('test.deflated', 'w');
/* Aqui "6" indica el nivel 6 de compresion */
stream_filter_append($da, 'zlib.deflate', STREAM_FILTER_WRITE, 6);
fwrite($da, $texto_original);
fclose($da);

echo "El archivo comprimido tiene " . filesize('test.deflated') . " bytes.\n";

/* Generates output:

El texto original tiene 70 caracteres.
El archivo comprimido tiene 56 bytes.

 */
?>
```

bzip2.compress y *bzip2.decompress* funcionan de la misma forma que los filtros zlib descritos anteriormente. El filtro *bzip2.compress* acepta hasta dos parámetros dados como elementos de una matriz asociativa: *blocks* es un valor entero desde 1 hasta 9 que indica el número de bloques de 100kbytes de memoria a reservar para el espacio de trabajo. *work* es también un valor entero que va desde 0 a 250, e indica cuánto esfuerzo debe invertirse para expandir usando el método de compresión normal antes de caer en un método más lento pero más confiable. Modificar este parámetro afecta únicamente la rapidez de compresién. Ni el tamaño de la salida comprimida ni el uso de memoria se modifican por este valor. Un factor de trabajo de 0 le indica a la biblioteca bzip que debe usar el valor interno predeterminado. El filtro *bzip2.decompress* sólo acepta un parámetro, el cual puede ser pasado como un valor booleano ordinario, o como el elemento *small* de una matriz asociativa. *small*, cuando se define a un valor **TRUE**, le indica a la biblioteca bzip de realice una descompresión con una cantidad de memoria mínima, al costo de la repidez.

Las filtros de compresión bzip2.* se encuentran disponibles con PHP desde la versión *5.1.0* si el soporte Referencia VII, *Funciones de compresión Bzip2* se encuentra habilitado. También se encuentran disponibles como una característica portada hacia atrás en la versión *5.0.x*, instalando el paquete bz2_filter desde PECL. Estos filtros *no* se encuentran disponibles para PHP 4.

Ejemplo M-9. *bzip2.compress* y *bzip2.decompress*

```
<?php
$param = array('blocks' => 9, 'work' => 0);

echo "El archivo original tiene " . filesize('LICENSE') . " bytes.\n";

$da = fopen('LICENSE.compressed', 'w');
stream_filter_append($da, 'bzip2.compress', STREAM_FILTER_WRITE, $param);
fwrite($da, file_get_contents('LICENSE'));
fclose($da);

echo "El archivo comprimido tiene " . filesize('LICENSE.compressed') . " bytes.\n";

/* Genera la salida:

El archivo original tiene 3288 bytes.
El archivo comprimido tiene 1488 bytes.

 */
?>
```

Apéndice N. Lista de Transportes de Sockets Soportados

La siguiente es una lista de los varios transportes de socket estilo URL que PHP tiene integrados para su uso con funciones de sockets basados en secuencias, tales como **fsockopen()**, y **stream_socket_client()**. Estos transportes *no* se aplican a la Extensión de Sockets.

Para una lista de transportes instalados con su versión de PHP use **stream_get_transports()**.

Dominio de Internet: TCP, UDP, SSL, y TLS

PHP 3, PHP 4, PHP 5. *ssl://* y *tls://* a partir de PHP 4.3.0, *sslv2://* y *sslv3://* a partir de PHP 5.0.2

> **Nota:** Si no se especifica un transporte, se asumirá *tcp://*

- *127.0.0.1*
- *fe80::1*
- *www.example.com*
- *tcp://127.0.0.1*
- *tcp://fe80::1*
- *tcp://www.example.com*
- *udp://www.example.com*
- *ssl://www.example.com*
- *sslv2://www.example.com*
- *sslv3://www.example.com*
- *tls://www.example.com*

Los sockets del Dominio de Internet esperan un número de puerto junto con una dirección de destino. En el caso de **fsockopen()**, éste es especificado en un segundo parámetro y por lo tanto no tiene impacto sobre el formato de la URL de transporte. Sin embargo, en el caso de **stream_socket_client()** y funciones relacionadas, así como ocurre con URLs tradicionales, el número de puerto se especifica como un sufijo del URL de transporte delimitado con el signo dos puntos.

- *tcp://127.0.0.1:80*
- *tcp://[fe80::1]:80*
- *tcp://www.example.com:80*

Direcciones IPv6 numéricas con números de puerto: En el segundo ejemplo anterior, mientras que los ejemplos IPv4 y con nombre de dominio fueron modificados solo ligeramente con la adición de sus dos puntos y número de puerto, la dirección IPv6 es rodeada por corchetes cuadrados: *[fe80::1]*. Esto es para poder distinguir entre los dos puntos usados en una dirección IPv6 y aquellos usados para delimitar el número de puerto.

Los transportes *ssl://* y *tls://* (disponibles únicamente cuando se compila el soporte para openssl con PHP) son extensiones del transporte *tcp://* el cual incluye encripción SSL. A partir de PHP 4.3.0 el soporte OpenSSL debe ser compilado estáticamente con PHP, a partir de PHP 5.0.0 puede ser compilado como módulo o estáticamente.

ssl:// intentará negociar una conexión SSL V2 o SSL V3 dependiendo de las capacidades y preferencias del host remoto. *sslv2://* y *sslv3://* seleccionarán el protocolo SSL V2 o SSL V3 explícitamente.

Tabla N-1. Opciones de contexto para los transportes *ssl://* y *tls://* (a partir de PHP 4.3.2)

Nombre	Uso	Predeterminado
verify_peer	**TRUE** o **FALSE**. Requerir verificación del certificado SSL usado.	**FALSE**
allow_self_signed	**TRUE** o **FALSE**. Permitir certificados firmados por uno mismo.	**FALSE**
cafile	Ubicación del archivo de Autoridad de Certificado en un sistema de archivos local, el cual debe ser usado con la opción de contexto *verify_peer* para verificar la identidad del conector remoto.	
capath	Si no se especifica *cafile* o el certificado no se encuentra allí, el directorio apuntado por *capath* es usado para buscar un certificado apropiado. *capath* debe ser un directorio de certificados correctamente habilitado mediante resumen criptográfico.	
local_cert	Ruta al archivo de certificado local en el sistema de archivos. Debe ser un archivo codificado mediante PEM, el cual contenga su certificado y llave privada. Opcionalmente, puede contener la cadena de expendedores del certificado.	
passphrase	La contraseña con la que el archivo *local_cert* fue codificado.	
CN_match	El nombre común (Common Name) que estamos esperando. PHP realizará comparaciones limitadas de comodines. Si el nombre común no produce coincidencias, el intento de conexión fallará.	

Nota: Debido a que *ssl://* es el transporte base para las envolturas *https://* y *ftps://*, cualquier opción de contexto que se aplique a *ssl://* también se aplica a *https://* y *ftps://*.

Dominio Unix: Unix y UDG

unix:// a partir de PHP 3, *udg://* a partir de PHP 5

- *unix:///tmp/misock*
- *udg:///tmp/misock*

unix:// provee acceso a una conexión secuencial de sockets en el dominio Unix. *udg://* provee un transporte alternativo a un socket del dominio Unix, usando el protocolo de datagramas de usuario.

Los sockets de dominio Unix, a diferencia de los sockets del dominio de Internet, no esperan un número de puerto. En el caso de **fsockopen()**, el parámetro *num_puerto* debe ser definido como 0.

Apéndice O. Tablas de comparación de tipos PHP

Las siguientes tablas demuestran los comportamientos de los tipos en PHP y los operadores de comparación, tanto para comparaciones flexibles como estrictas. Este suplemento se encuentra relacionado también con la sección del manual sobre manipulación de tipos. La inspiración ha provenido de varios comentarios de usuarios, y del trabajo realizado en BlueShoes.

Antes de usar estas tablas, es importante entender los tipos y sus significados. Por ejemplo, *"42"* es **string** mientras que *42* es un **integer**. **FALSE** es un **boolean** mientras *"false"* es **string**.

> **Nota:** Los Formularios HTML no pasan enteros, reales, o valores booleanos; ellos pasan cadenas. Para saber si una cadena es numérica, usted puede usar **is_numeric()**.

> **Nota:** Hacer un simple *if ($x)* cuando *$x* no esté definido, generará un error de nivel **E_NOTICE**. En lugar de esto, considere el uso de **empty()** o **isset()**, o inicializar sus variables.

Tabla O-1. Comparaciones de *$x* con funciones PHP

Expresión	gettype()	empty()	is_null()	isset()	boolean : *if ($x)*
$x = "";	string	TRUE	FALSE	TRUE	FALSE
$x = ***NULL***	NULL	TRUE	TRUE	FALSE	FALSE
var $x;	NULL	TRUE	TRUE	FALSE	FALSE
$x no se encuentra definida	NULL	TRUE	TRUE	FALSE	FALSE
$x = array();	array	TRUE	FALSE	TRUE	FALSE
$x = false;	boolean	TRUE	FALSE	TRUE	FALSE
$x = true;	boolean	FALSE	FALSE	TRUE	TRUE
$x = 1;	integer	FALSE	FALSE	TRUE	TRUE
$x = 42;	integer	FALSE	FALSE	TRUE	TRUE
$x = 0;	integer	TRUE	FALSE	TRUE	FALSE
$x = -1;	integer	FALSE	FALSE	TRUE	TRUE
$x = "1";	string	FALSE	FALSE	TRUE	TRUE
$x = "0";	string	TRUE	FALSE	TRUE	FALSE
$x = "-1";	string	FALSE	FALSE	TRUE	TRUE
$x = "php";	string	FALSE	FALSE	TRUE	TRUE

Expresión	gettype()	empty()	is_null()	isset()	boolean : *if ($x)*
$x = "true";	string	FALSE	FALSE	TRUE	TRUE
$x = "false";	string	FALSE	FALSE	TRUE	TRUE

Tabla O-2. Comparaciones flexibles con ==

	TRUE	FALSE	*1*	*0*	*-1*	*"1"*	*"0"*	*"-1"*	NULL	*array()*	*"php"*
TRUE	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE
FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE
1	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE
0	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE
-1	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE
"1"	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE
"0"	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
"-1"	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE
NULL	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE
array()	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE
"php"	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE

Tabla O-3. Comparaciones estrictas con ===

	TRUE	FALSE	*1*	*0*	*-1*	*"1"*	*"0"*	*"-1"*	NULL	*array()*	*"php"*
TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
1	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
0	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
-1	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
"1"	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE
"0"	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
"-1"	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE
NULL	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
array()	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE
"php"	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE

Nota de PHP 3.0: El valor de cadena *"0"* fue considerado como no-vacío en PHP 3, este comportamiento cambió en PHP 4 en donde es visto ahora como vacío.

Apéndice P. Lista de Identificadores (tokens) del Analizador

Varias partes del lenguaje PHP están representadas internamente por cosas tales como T_SR. El PHP muestra identificadores como éste en los errores durante el análisis gramatical, como por

ejemplo: "Parse error: unexpected T_SR, expecting ',' or ';' in script.php on line 10." ("Error de análisis: T_SR inesperado, se esperaba ',' o ';' en la línea 10 de script.php.").

Suponemos que sabeis que significa T_SR. Para quienes no lo conocen, aquí hay una tabla con esos identificadores, la sintaxis de PHP, y referencias a lugares apropiados del manual.

Tabla P-1. Identificadores (Tokens)

Identificador	Sintaxis	Referencia
T_AND_EQUAL	&=	Operadores de Asignación
T_ARRAY	array()	**array()**, Sintaxis de array
T_ARRAY_CAST	(array)	Forzado de Tipos
T_AS	as	foreach
T_BAD_CHARACTER		cualquier caracter debajo del ASCII 32, excepto \t (0x09), \n (0x0a) y \r (0x0d)
T_BOOLEAN_AND	&&	Operadores Lógicos
T_BOOLEAN_OR	||	Operadores Lógicos
T_BOOL_CAST	(bool) o (boolean)	Forzado de Tipos
T_BREAK	break	break
T_CASE	case	switch
T_CHARACTER		
T_CLASS	class	Clases y Objetos
T_CLOSE_TAG	?> o %>	
T_COMMENT	// o #	Comentarios
T_CONCAT_EQUAL	.=	Operadores de Asignación
T_CONST	const	
T_CONSTANT_ENCAPSED_STRING	"foo" o 'bar'	Sintasis de Cadenas
T_CONTINUE	continue	
T_CURLY_OPEN		
T_DEC	--	Operadores de Incremento/decremento
T_DECLARE	declare	declare
T_DEFAULT	default	switch
T_DIV_EQUAL	/=	Operadores de Asignación
T_DNUMBER	0.12, etc.	Números en Punto Flotante
T_DO	do	do..while
T_DOLLAR_OPEN_CURLY_BRACES	${	Sintaxis de Variables Complejas Analizadas
T_DOUBLE_ARROW	=>	Sintaxis de Matrices
T_DOUBLE_CAST	(real), (double) or (float)	Forzado de Tipos
T_ECHO	echo	**echo()**

Identificador	Sintaxis	Referencia
T_ELSE	else	else
T_ELSEIF	elseif	elseif
T_EMPTY	empty	**empty()**
T_ENCAPSED_AND_WHITESPACE		
T_ENDDECLARE	enddeclare	declare, Sintaxis Alternativa
T_ENDFOR	endfor	for, Sintaxis Alternativa
T_ENDFOREACH	endforeach	foreach, Sintaxis Alternativa
T_ENDIF	endif	if, Sintaxis Alternativa
T_ENDSWITCH	endswitch	switch, Sintaxis Alternativa
T_ENDWHILE	endwhile	while, Sintaxis Alternativa
T_END_HEREDOC		heredoc
T_EVAL	eval()	**eval()**
T_EXIT	exit o die	**exit()**, **die()**
T_EXTENDS	extends	extends, Clases y Objetos
T_FILE	__FILE__	Constantes
T_FOR	for	for
T_FOREACH	foreach	foreach
T_FUNCTION	function o cfunction	Funciones
T_GLOBAL	global	Ambito de Variables
T_IF	if	if
T_INC	++	Operadores de Incremento/decremento
T_INCLUDE	include()	**include()**
T_INCLUDE_ONCE	include_once()	**include_once()**
T_INLINE_HTML		
T_INT_CAST	(int) o (integer)	Forzado de Tipos
T_ISSET	isset()	**isset()**
T_IS_EQUAL	==	Operadores de Comparación
T_IS_GREATER_OR_EQUAL	>=	Operadores de Comparación
T_IS_IDENTICAL	===	Operadores de Comparación
T_IS_NOT_EQUAL	!= o <>	Operadores de Comparación
T_IS_NOT_IDENTICAL	!==	Operadores de Comparación
T_SMALLER_OR_EQUAL	<=	Operadores de Comparación
T_LINE	__LINE__	Constantes
T_LIST	list()	**list()**
T_LNUMBER	123, 012, 0x1ac, etc.	Enteros

Identificador	Sintaxis	Referencia
T_LOGICAL_AND	and	Operadores Lógicos
T_LOGICAL_OR	or	Operadores Lógicos
T_LOGICAL_XOR	xor	Operadores Lógicos
T_MINUS_EQUAL	-=	Operadores de Asignación
T_ML_COMMENT	/* y */	Comentarios
T_MOD_EQUAL	%=	Operadores de Asignación
T_MUL_EQUAL	*=	Operadores de Asignación
T_NEW	new	Clases y Objetos
T_NUM_STRING		
T_OBJECT_CAST	(object)	Forzado de Tipos
T_OBJECT_OPERATOR	->	Clases y Objetos
T_OLD_FUNCTION	old_function	old_function
T_OPEN_TAG	<?php, <? o <%	Saliendo de HTML
T_OPEN_TAG_WITH_ECHO	<?= o <%=	Saliendo de HTML
T_OR_EQUAL	|=	Operadores de Asignación
T_PAAMAYIM_NEKUDOTAYIM	::	..
T_PLUS_EQUAL	+=	Operadores de Asignación
T_PRINT	print()	**print()**
T_REQUIRE	require()	**require()**
T_REQUIRE_ONCE	require_once()	**require_once()**
T_RETURN	return	Retorno de Valores
T_SL	<<	Operadores a Nivel de Bits
T_SL_EQUAL	<<=	Operadores de Asignación
T_SR	>>	Operadores a Nivel de Bits
T_SR_EQUAL	>>=	Operadores de Asignación
T_START_HEREDOC	<<<	heredoc
T_STATIC	static	Ambito de las Variables
T_STRING		
T_STRING_CAST	(string)	Forzado de Variables
T_STRING_VARNAME		
T_SWITCH	switch	switch
T_UNSET	unset()	**unset()**
T_UNSET_CAST	(unset)	(no documentado; arroja a **NULL**)
T_USE	use	(no implementado)
T_VAR	var	Clases y Objetos
T_VARIABLE	$foo	Variables
T_WHILE	while	while, do..while

Identificador	Sintaxis	Referencia
T_WHITESPACE		
T_XOR_EQUAL	^=	Operadores de Asignación
T_FUNC_C	__FUNCTION__	constants, desde *PHP* 4.3.0
T_CLASS_C	__CLASS__	constants, desde *PHP* 4.3.0

Apéndice Q. Sobre el manual

Formatos

El manual de PHP se encuentra disponible en diferentes formatos. Estos formatos se pueden dividir en dos grupos: formatos accesible en línea (online) y ficheros que se pueden bajar a tu ordenador.

> **Nota:** Algunos editores han publicado versiones impresas de este manual. No podemos recomendar ninguna de ellas, ya que suelen quedarse obsoletas rápidamente.

Podeis leer el manual en linea (online) en http://www.php.net/ y en numerosos servidores espejo. Para mejorar la respuesta, deberiais utilizar el servidor espejo más cercano a vosotros. Podeis consultar el manual en formato HTML -impresión amigable- ó en formato HTML que integra el manual en el diseno de la página web del proyecto PHP.

Una ventaja del manual en línea sobre la mayoria de formatos fuera de línea es la integración de los comentarios/notas de usuarios. Una desventaja es que teneis que estar conectados para ver consultar las versiones en línea.

Existen varios formatos fuera de línea y el más apropiado para cada usuario depende del sistema operativo que se use y de tus preferencias a la hora de leer. Informacion sobre como el manual es generado en tantos formatos diferentes se puede obtener en la seccion de este apendice titulada 'Como generamos los dierentes formatos'.

Los formatos más accesibles desde diferentes plataformas son HTML y texto plano. el manual en HTML se proporciona como un unico fichero en HTML ó como un paquete con un fichero HTML por sección. Los formatos HTML y texto plano se distribuyen como ficheros tar y con compresión bz2.

Otro formato muy popular y el más adecuado para imprimir, es PDF (Adobe Acrobat). Tenemos que advertir antes que empeceis a imprimir que el manual contiene casi 2000 páginas y se revisa constantemente.

> **Nota:** Si no teneis un programa capaz de presentar ficheros PDF, podeis bajaros Adobe Acrobat Reader.

Los usuarios de ordenadores de mano compatibles con Palm, tienen los formatos Palm doc y iSilo. Podreis utlizar un lector de documentos DOC ó iSilo para consultar la documentación de PHP o como referencia rápida.

Para plataformas Windows, la versión Windows HTML Ayuda proporciona el formato HTML usado con la aplicacion de ayuda Windows HTML. Esta versión proporciona busqueda completa de texto, indice completo y marcas (bookmarks). Muchos entornos de desarrollo para PHP en Windows integran esta versión.

Sobre las notas de usuarios

Las notas de usuarios juegan un importante papael en el desarrollo de este manual. Permitiendo a los lectores del manual contribuir con ejemplos y aclaraciones desde su navegador, podemos incorporar estas aclaraciones en el manual. Hasta que las notas son incorporadas en el manual, pueden ser accedidas desde el manual en línea y algunos formatos fuera de línea.

> **Nota:** Las notas de usuario no son moderadas antes de estar disponibles en línea, asi que la calidad de lo escrito asi como los ejemplos ó incluso la veracidad de lo expuesto en la nota no puede ser garantizada.

> **Nota:** Las notas contribuidas por los usuarios son consideradas parte del manual de PHP. Por lo tanto estan cubiertas por la misma licencia que se aplica a esta documentacion (Por el momento GPL). Para mas detalles, podeis consultar el Copyright del manual.

Como interpretar la definición de una función (prototipo)

Cada función está documentada para una obtener una referencia rápida, si sabemos como interpretar y entender el manual podremos usar PHP de una manera más sencilla. En vez de confiar en ejemplos ó cortar y pegar, podeis aprender como interpretar las definiciones de funciones. Empecemos:

> **Prerequisitos: Comprensión básica de los diferentes tipos:** Aunque PHP es un lenguaje de programación relajado en lo referente a los tipos de variables/valores, es importante entender básicamente los diferentes tipos, ya que son importantes.

Las definiciones de funciones nos dicen que tipo de valores es retornado. Usemos la definición de la función **strlen()** en nuestro primer ejemplo:

```
strlen

(PHP 3, PHP 4 >= 4.0.0)
strlen -- Obtiene la longitud de una cadena.

Descripción
int strlen ( string str )

Retorna la longitud de una cadena.
```

Tabla Q-1. Explicación de la definición de una función

Parte	Descripción
strlen	El nombre de la función.
(PHP 3, PHP 4 >= 4.0.0)	strlen() está disponible tanto en PHP 3 como en PHP 4
int	Tipo de valor devuelto por esta función, en este caso un entero (La longitud de una cadena es medida en numeros)
(string str)	El primer (y en este caso el único) parámetro/argumento de la función strlen() se llama *str* y es una cadena.

Podriamos escribir la definicion de esta función de un modo genérico:

```
Tipo de valor retornado - nombre de función (tipo de parametro - nombre del parametro)
```

Muchas funciones tiene multiples parametros, por ejemplo **in_array()**. Su definición seria:

```
bool in_array ( mixed needle, array haystack [, bool strict])
```

Esto significa lo siguiente, in_array() retorna un valor boolean, **TRUE** si termina con exito (el parametro *needle* fue encontrado en la matriz *haystack*) ó **FALSE** si falla (el parametro *needle* no fue encontrado en la matriz *haystack*). El primer parametro se llama *needle* y puede tener valores de diferente tipo, asi que lo llamamos de tipo "*mixed*". Este parametro *needle* (lo que estamos buscando) puede ser un valor escalar (cadena (string), entero (integer) o flotante(float)), ó una matriz. *haystack*, (la matriz en la que estamos buscando) es el segundo parámetro. El tercer parametro *opcional* se llama *strict*. Todos los parámetros opcionales se encuentran entre *[* corchetes *]*. El manual define que el parametro *strict* por defecto retorna **FALSE**. Consultar el manual para obtener detalles de como las diferentes funciones funcionan.

Versiones de PHP documentadas en este manual

Esta documentación contiene informacion sobre PHP 4, con algunas notas adicionales sobre migración y compatibilidad con PHP 3. Comportamientos, parámetros, valores retornados y otros cambios entre diferentes versiones de PHP se encuentran documentados en notas en este manual.

Podeis encontrar trozos de documentación para la versión CVS de PHP, esto significa la última versión de desarrollo disponible por el momento. Si no eres un desarrollador de PHP ó no utilizas la última versión de PHP en el CVS, no tendras disponibles las características marcadas con "disponible en el CVS". De todas maneras, estas características estarán probablemente disponibles en la proxima versión estable de PHP. Si deseais bajaros la versión del CVS, consultar la página sobre acceso anónimo al CVS.

También podeis encontrar documentación sobre una versión de PHP que no esté publicada todavia (p.ej PHP 5.0.0 cuando la última versión estable es 4.3.0). La mayoria de las veces no es un fallo. La explicación es añadida para características que se sabe estarán disponibles en una versión futura de PHP.

Como encontrar más información sobre PHP

Este manual no pretende explicar practicas generales de programación. Si vas a empezar a programar ó eres un programador principiante, seguramente tendras dificultades para aprender a programar en PHP usando solamente este manual. Te será de gran ayuda utilizar otros textos más orientados a principiantes. Una lista con libros sobre PHP puede encontrarse en http://www.php.net/books.php.

Existen una serie de listas de correo en donde se puede discutir sobre diferentes aspectos relacionados con la programación en PHP. Si tienes un problema, el cual no puedes solucionar, puedes usar estas listas para preguntar y obtener ayuda. Una lista con las listas de correo disponibles puede encontrarse en http://www.php.net/support.php, asi como enlaces a los archivos historicos de estas listas y otros recursos de ayuda en línea. Además en http://www.php.net/links.php puedes encontrar información sobre servidores web dedicados a PHP, artículos, foros y galerias de código.

Como ayudar a mejorar la documentación

Se pueden ayudar de tres maneras a mejorar esta documentción.

Si encontrais errores en este manual, en cualquier lenguaje, podeis informar sobre el mismo, usando el sistema de informes de errores en http://bugs.php.net/. Clasificar el error como "Documentation Problem". Problemas encontrados con los diferentes formatos tambien se pueden mandar usando este sistema.

> **Nota:** Por favor, no abusar del sistema de informes de errores mandando peticiones de ayuda. Utilizar las listas de correos y recursos mencionados anteriormente.

Contribuyendo con notas/comentarios, podeis dar ejemplos adicionales y aclaraciones a otros lectores. Pero no mandar informes de errores mediante el sistema de notas/comentarios. Podeis leer más sobre notas/comentarios en la sección 'Sobre las notas de usuarios' de este apendice.

Si sabes ingles y otro idioma, podeis ayudar en las traducciones. Si te interesa comenzar una nueva traducción o ayudar una existente, pasate por http://cvs.php.net/co.php/phpdoc/howto/howto.html.tar.gz.

Como generamos los formatos

This manual is written in XML using the DocBook XML DTD, using DSSSL (Document Style and Semantics Specification Language) for formatting, and experimentally the XSLT (Extensible Stylesheet Language Transformations) for maintenance and formatting.

Usando XML como código fuente, podemos generar diferentes formatos, manteniendo solamente un solo código fuente para todos los formatos. Las herramientas utilizadas para crear las versiones HTML y TeX son Jade, escrito por James Clark y The Modular DocBook Stylesheets, escrito por Norman Walsh. Usamos Microsoft HTML Help Workshop para generar el formato de ayuda Windows HTML y por supuesto PHP para algunas conversiones y preprocesado.

Puedes bajarte el manual en diferentes idiomas y formatos, incluyendo texto plano, HTML,PDF, PalmPilot DOC, PalmPilot iSilo y Windows HTML Help, de http://www.php.net/docs.php. Los manuales son generados automaticamente a medida que el código fuente se va actualizando.

Más información sobre como bajarse el código fuente en XML en http://cvs.php.net/. La documentación se encuentra en el módulo *phpdoc*

Traducciones

El manual de PHP no está disponible solamente en diferentes formatos, tambien se encuentra en diferentes idiomas. El manual es escrito primero en ingles y luego grupos de traductores de todo el mundo se encargan de mantener la traducción a su idioma sincronizada con la versión inglesa. Si la traducción de algún apartado del manual no ha sido realizada todavia por el equipo de traductores, este aparecerá en ingles hasta que sea traducido.

Las personas involucradas en las traducciones, utilizan el código fuente en XML disponible en http://cvs.php.net/ y lo traducen a su idioma. *No utilizan* la versión en HTML, texto plano ó PDF. Es el sistema de generación de formatos el que se encarga de producirlos a partir de XML.

> **Nota:** Si te interesa ayudar ó participar en la traducción de la documentación a tu idioma, ponte en contacto con el equipo de traducción/documentación apuntandote la la lista de correo phpdoc. Manda un mensaje vacio a phpdoc-subscribe@lists.php.net. La dirección de la esta lista de correo es *phpdoc@lists.php.net*. Explica que estás interesado en traducir el manual a un idioma y nos pondremos en contacto contigo, te mandaremos información para que puedas empezar una nueva traducción o para que pases a formar parte del equipo de traducción de tu idioma, si este ya existe.

Por el momento existen traducciones, completas o parciales, en los siguientes idiomas: Brasileño, Portugues, Chino (simplificado), Chino (Hong Kong Cantonese),Chino (Tradicional), Checo, Holandés, Finlandés, Frances, Alemán, Hebreo, Hungaro, Italiano, Japones, Koreano, Polaco, Rumano, Ruso, Eslovaco, Español, Sueco y Turco.

Todas ellas se pueden bajar de: http://www.php.net/docs.php.

Sobre la traducción al español

El proyecto de traducción de la documantación de PHP al español tiene una página web en la que podrás encontrar información sobre el mismo, historia, que se necesita para colaborar, guia para los colaboradores, reglas y convenciones de traducción, listas de correos, acceso a la documentación y a los colaboradores de la traducción, etc. La dirección es http://www.linux-es.com/PHP/

La traducción del manual de PHP al español ha sido posible gracias a la colaboración de un gran número de traductores, que desinteresadamente han usado su tiempo para que todos podamos tener una versión en nuestra lengua de esta documentación.

El coordinador de la traducción es Rafael Martinez podeis contactar con el, para cualquier duda relacionada con el proyecto de traducción. Los colaboradores los podeis encontrar en esta lista:

Apéndice R. Open Publication License

v1.0, 8 June 1999

I. REQUIREMENTS ON BOTH UNMODIFIED AND MODIFIED VERSIONS

The Open Publication works may be reproduced and distributed in whole or in part, in any medium physical or electronic, provided that the terms of this license are adhered to, and that this license or an incorporation of it by reference (with any options elected by the author(s) and/or publisher) is displayed in the reproduction.

Proper form for an incorporation by reference is as follows:

Copyright (c) <year> by <author's name or designee>. This material may be distributed only subject to the terms and conditions set forth in the Open Publication License, vX.Y or later (the latest version is presently available at http://www.opencontent.org/openpub/

The reference must be immediately followed with any options elected by the author(s) and/or publisher of the document (see section VI). Commercial redistribution of Open Publication-licensed material is permitted. Any publication in standard (paper) book form shall require the citation of the original publisher and author. The publisher and author's names shall appear on all outer surfaces of the book. On all outer surfaces of the book the original publisher's name shall be as large as the title of the work and cited as possessive with respect to the title.

II. COPYRIGHT

The copyright to each Open Publication is owned by its author(s) or designee.

III. SCOPE OF LICENSE

The following license terms apply to all Open Publication works, unless otherwise explicitly stated in the document.

Mere aggregation of Open Publication works or a portion of an Open Publication work with other works or programs on the same media shall not cause this license to apply to those other works. The aggregate work shall contain a notice specifying the inclusion of the Open Publication material and appropriate copyright notice.

SEVERABILITY. If any part of this license is found to be unenforceable in any jurisdiction, the remaining portions of the license remain in force.

NO WARRANTY. Open Publication works are licensed and provided "as is" without warranty of any kind, express or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose or a warranty of non-infringement.

IV. REQUIREMENTS ON MODIFIED WORKS

All modified versions of documents covered by this license, including translations, anthologies, compilations and partial documents, must meet the following requirements:

1. The modified version must be labeled as such.

2. The person making the modifications must be identified and the modifications dated.

3. Acknowledgement of the original author and publisher if applicable must be retained according to normal academic citation practices.

4. The location of the original unmodified document must be identified.

5. The original author's (or authors') name(s) may not be used to assert or imply endorsement of the resulting document without the original author's (or authors') permission.

V. GOOD-PRACTICE RECOMMENDATIONS

In addition to the requirements of this license, it is requested from and strongly recommended of redistributors that:

1. If you are distributing Open Publication works on hardcopy or CD-ROM, you provide email notification to the authors of your intent to redistribute at least thirty days before your manuscript or media freeze, to give the authors time to provide updated documents. This notification should describe modifications, if any, made to the document.

2. All substantive modifications (including deletions) be either clearly marked up in the document or else described in an attachment to the document.

3. Finally, while it is not mandatory under this license, it is considered good form to offer a free copy of any hardcopy and CD-ROM expression of an Open Publication-licensed work to its author(s).

VI. LICENSE OPTIONS

The author(s) and/or publisher of an Open Publication-licensed document may elect certain options by appending language to the reference to or copy of the license. These options are considered part of the license instance and must be included with the license (or its incorporation by reference) in derived works.

A. To prohibit distribution of substantively modified versions without the explicit permission of the author(s). "Substantive modification" is defined as a change to the semantic content of the document, and excludes mere changes in format or typographical corrections.

To accomplish this, add the phrase `Distribution of substantively modified versions of this document is prohibited without the explicit permission of the copyright holder.' to the license reference or copy.

B. To prohibit any publication of this work or derivative works in whole or in part in standard (paper) book form for commercial purposes is prohibited unless prior permission is obtained from the copyright holder.

To accomplish this, add the phrase 'Distribution of the work or derivative of the work in any standard (paper) book form is prohibited unless prior permission is obtained from the copyright holder.' to the license reference or copy.

Apéndice S. Índice de funciones

Índice de funciones

A

array_fill()
array_filter()
array_flip()
array_intersect()
array_intersect_assoc()
array_intersect_key()
array_intersect_uassoc()
array_intersect_ukey()
array_key_exists()
array_keys()
array_map()
array_merge()
array_merge_recursive()
array_multisort()
array_pad()
array_pop()
array_push()
array_rand()
array_reduce()
array_reverse()
array_search()
array_shift()
array_slice()
array_splice()
array_sum()
array_udiff()
array_udiff_assoc()
array_udiff_uassoc()
array_uintersect()
array_uintersect_assoc()
array_uintersect_uassoc()
array_unique()
array_unshift()
array_values()
array_walk()
array_walk_recursive()
ArrayIterator::current()
ArrayIterator::key()
ArrayIterator::next()
ArrayIterator::rewind()
ArrayIterator::seek()
ArrayIterator::valid()
ArrayObject::__construct()
ArrayObject::append()
ArrayObject::count()
ArrayObject::getIterator()
ArrayObject::offsetExists()
ArrayObject::offsetGet()
ArrayObject::offsetSet()
ArrayObject::offsetUnset()
arsort()
ascii2ebcdic()
asin()
asinh()

B

C

D

descriptor->free()
dgettext()
die()
dio_close()
dio_fcntl()
dio_open()
dio_read()
dio_seek()
dio_stat()
dio_tcsetattr()
dio_truncate()
dio_write()
dir()
DirectoryIterator::__construct()
DirectoryIterator::current()
DirectoryIterator::getATime()
DirectoryIterator::getChildren()
DirectoryIterator::getCTime()
DirectoryIterator::getFilename()
DirectoryIterator::getGroup()
DirectoryIterator::getInode()
DirectoryIterator::getMTime()
DirectoryIterator::getOwner()
DirectoryIterator::getPath()
DirectoryIterator::getPathname()
DirectoryIterator::getPerms()
DirectoryIterator::getSize()
DirectoryIterator::getType()
DirectoryIterator::isDir()
DirectoryIterator::isDot()
DirectoryIterator::isExecutable()
DirectoryIterator::isFile()
DirectoryIterator::isLink()
DirectoryIterator::isReadable()
DirectoryIterator::isWritable()
DirectoryIterator::key()
DirectoryIterator::next()
DirectoryIterator::rewind()
DirectoryIterator::valid()
dirname()
disk_free_space()
disk_total_space()
diskfreespace()
dl()
dngettext()
dns_check_record()
dns_get_mx()
dns_get_record()
dom_import_simplexml()
DOMAttr->isId()
DomAttribute->name()
DomAttribute->specified()
DomAttribute->value()
DOMCharacterData->appendData()

DOMCharacterData->deleteData()
DOMCharacterData->insertData()
DOMCharacterData->replaceData()
DOMCharacterData->substringData()
DOMDocument->__construct()
DomDocument->add_root()
DomDocument->create_attribute()
DomDocument->create_cdata_section()
DomDocument->create_comment()
DomDocument->create_element()
DomDocument->create_element_ns()
DomDocument->create_entity_reference()
DomDocument->create_processing_instruction()
DomDocument->create_text_node()
DOMDocument->createAttribute()
DOMDocument->createAttributeNS()
DOMDocument->createCDATASection()
DOMDocument->createComment()
DOMDocument->createDocumentFragment()
DOMDocument->createElement()
DOMDocument->createElementNS()
DOMDocument->createEntityReference()
DOMDocument->createProcessingInstruction()
DOMDocument->createTextNode()
DomDocument->doctype()
DomDocument->document_element()
DomDocument->dump_file()
DomDocument->dump_mem()
DomDocument->get_element_by_id()
DomDocument->get_elements_by_tagname()
DOMDocument->getElementById()
DOMDocument->getElementsByTagName()
DOMDocument->getElementsByTagNameNS()()
DomDocument->html_dump_mem()
DOMDocument->importNode()
DOMDocument->load()
DOMDocument->loadHTML()
DOMDocument->loadHTMLFile()
DOMDocument->loadXML()
DOMDocument->normalize()
DOMDocument->relaxNGValidate()
DOMDocument->relaxNGValidateSource()
DOMDocument->save()
DOMDocument->saveHTML()()
DOMDocument->saveHTMLFile()()
DOMDocument->saveXML()()
DOMDocument->schemaValidate()
DOMDocument->schemaValidateSource()
DOMDocument->validate()
DOMDocument->xinclude()
DomDocument->xinclude()
DomDocumentType->entities()
DomDocumentType->internal_subset()
DomDocumentType->name()

DomDocumentType->notations()
DomDocumentType->public_id()
DomDocumentType->system_id()
DomElement->get_attribute()
DomElement->get_attribute_node()
DomElement->get_elements_by_tagname()
DOMElement->getAttribute()
DOMElement->getAttributeNode()
DOMElement->getAttributeNodeNS()
DOMElement->getAttributeNS()
DOMElement->getElementsByTagName()
DOMElement->getElementsByTagNameNS()
DomElement->has_attribute()
DOMElement->hasAttribute()
DOMElement->hasAttributeNS()
DomElement->remove_attribute()
DOMElement->removeAttribute()
DOMElement->removeAttributeNode()
DOMElement->removeAttributeNS()
DomElement->set_attribute()
DOMElement->setAttribute()
DOMElement->setAttributeNode()
DOMElement->setAttributeNodeNS()
DOMElement->setAttributeNS()
DomElement->tagname()
DOMImplementation->createDocument()
DOMImplementation->createDocumentType()
DOMImplementation->hasFeature()()
DOMNamedNodeMap->getNamedItem()
DOMNamedNodeMap->getNamedItemNS()
DOMNamedNodeMap->item()
DomNode->add_namespace()
DomNode->append_child()
DomNode->append_sibling()
DOMNode->appendChild()
DomNode->attributes()
DomNode->child_nodes()
DomNode->clone_node()
DOMNode->cloneNode()
DomNode->dump_node()
DomNode->first_child()
DomNode->get_content()
DomNode->has_attributes()
DomNode->has_child_nodes()
DOMNode->hasAttributes()
DOMNode->hasChildNodes()
DomNode->insert_before()
DOMNode->insertBefore()
DomNode->is_blank_node()
DOMNode->isSameNode()
DOMNode->isSupported()
DomNode->last_child()
DOMNode->lookupNamespaceURI()
DOMNode->lookupPrefix()

DomNode->next_sibling()
DomNode->node_name()
DomNode->node_type()
DomNode->node_value()
DOMNode->normalize()
DomNode->owner_document()
DomNode->parent_node()
DomNode->prefix()
DomNode->previous_sibling()
DomNode->remove_child()
DOMNode->removeChild()
DomNode->replace_child()
DomNode->replace_node()
DOMNode->replaceChild()
DomNode->set_content()
DomNode->set_name()
DomNode->set_namespace()
DomNode->unlink_node()
DOMNodelist->item()
DomProcessingInstruction->data()
DomProcessingInstruction->target()
DOMStringList->item()
DOMText->isWhitespaceInElementContent()
DOMText->splitText()
domxml_new_doc()
domxml_open_file()
domxml_open_mem()
domxml_version()
domxml_xmltree()
domxml_xslt_stylesheet()
domxml_xslt_stylesheet_doc()
domxml_xslt_stylesheet_file()
DOMXPath->__construct()
DOMXPath->evaluate()()
DOMXPath->query()
DOMXPath->registerNamespace()
DomXsltStylesheet->process()
DomXsltStylesheet->result_dump_file()
DomXsltStylesheet->result_dump_mem()
dotnet()
dotnet_load()
doubleval()

E

each()
easter_date()
easter_days()
ebcdic2ascii()
echo()
empty()
end()

F

fdf_get_value()
fdf_get_version()
fdf_header()
fdf_next_field_name()
fdf_open()
fdf_open_string()
fdf_remove_item()
fdf_save()
fdf_save_string()
fdf_set_ap()
fdf_set_encoding()
fdf_set_file()
fdf_set_flags()
fdf_set_javascript_action()
fdf_set_on_import_javascript()
fdf_set_opt()
fdf_set_status()
fdf_set_submit_form_action()
fdf_set_target_frame()
fdf_set_value()
fdf_set_version()
feof()
fflush()
fgetc()
fgetcsv()
fgets()
fgetss()
file()
file_exists()
file_get_contents()
file_put_contents()
fileatime()
filectime()
filegroup()
fileinode()
filemtime()
fileowner()
fileperms()
filepro()
filepro_fieldcount()
filepro_fieldname()
filepro_fieldtype()
filepro_fieldwidth()
filepro_retrieve()
filepro_rowcount()
filesize()
filetype()
FilterIterator::current()
FilterIterator::getInnerIterator()
FilterIterator::key()
FilterIterator::next()
FilterIterator::rewind()
FilterIterator::valid()
floatval()

G

H

hw_api->parents()
hw_api->remove()
hw_api->replace()
hw_api->setcommittedversion()
hw_api->srcanchors()
hw_api->srcsofdst()
hw_api->unlock()
hw_api->user()
hw_api->userlist()
hw_api_attribute()
hw_api_attribute->key()
hw_api_attribute->langdepvalue()
hw_api_attribute->value()
hw_api_attribute->values()
hw_api_content()
hw_api_content->mimetype()
hw_api_content->read()
hw_api_error->count()
hw_api_error->reason()
hw_api_object()
hw_api_object->assign()
hw_api_object->attreditable()
hw_api_object->count()
hw_api_object->insert()
hw_api_object->remove()
hw_api_object->title()
hw_api_object->value()
hw_api_reason->description()
hw_api_reason->type()
hw_array2objrec()
hw_changeobject()
hw_children()
hw_childrenobj()
hw_close()
hw_connect()
hw_connection_info()
hw_cp()
hw_deleteobject()
hw_docbyanchor()
hw_docbyanchorobj()
hw_document_attributes()
hw_document_bodytag()
hw_document_content()
hw_document_setcontent()
hw_document_size()
hw_dummy()
hw_edittext()
hw_error()
hw_errormsg()
hw_free_document()
hw_getanchors()
hw_getanchorsobj()
hw_getandlock()
hw_getchildcoll()

I

imagecolorat()
imagecolorclosest()
imagecolorclosestalpha()
imagecolorclosesthwb()
imagecolordeallocate()
imagecolorexact()
imagecolorexactalpha()
imagecolormatch()
imagecolorresolve()
imagecolorresolvealpha()
imagecolorset()
imagecolorsforindex()
imagecolorstotal()
imagecolortransparent()
imagecopy()
imagecopymerge()
imagecopymergegray()
imagecopyresampled()
imagecopyresized()
imagecreate()
imagecreatefromgd()
imagecreatefromgd2()
imagecreatefromgd2part()
imagecreatefromgif()
imagecreatefromjpeg()
imagecreatefrompng()
imagecreatefromstring()
imagecreatefromwbmp()
imagecreatefromxbm()
imagecreatefromxpm()
imagecreatetruecolor()
imagedashedline()
imagedestroy()
imageellipse()
imagefill()
imagefilledarc()
imagefilledellipse()
imagefilledpolygon()
imagefilledrectangle()
imagefilltoborder()
imagefilter()
imagefontheight()
imagefontwidth()
imageftbbox()
imagefttext()
imagegammacorrect()
imagegd()
imagegd2()
imagegif()
imageinterlace()
imageistruecolor()
imagejpeg()
imagelayereffect()
imageline()

J

K

L

M

mcve_numrows()
mcve_override()
mcve_parsecommadelimited()
mcve_ping()
mcve_preauth()
mcve_preauthcompletion()
mcve_qc()
mcve_responseparam()
mcve_return()
mcve_returncode()
mcve_returnstatus()
mcve_sale()
mcve_setblocking()
mcve_setdropfile()
mcve_setip()
mcve_setssl()
mcve_setssl_files()
mcve_settimeout()
mcve_settle()
mcve_text_avs()
mcve_text_code()
mcve_text_cv()
mcve_transactionauth()
mcve_transactionavs()
mcve_transactionbatch()
mcve_transactioncv()
mcve_transactionid()
mcve_transactionitem()
mcve_transactionssent()
mcve_transactiontext()
mcve_transinqueue()
mcve_transnew()
mcve_transparam()
mcve_transsend()
mcve_ub()
mcve_uwait()
mcve_verifyconnection()
mcve_verifysslcert()
mcve_void()
md5()
md5_file()
mdecrypt_generic()
Memcache::add()
Memcache::close()
Memcache::connect()
Memcache::decrement()
Memcache::delete()
Memcache::flush()
Memcache::get()
Memcache::getStats()
Memcache::getVersion()
Memcache::increment()
Memcache::pconnect()
Memcache::replace()

Memcache::set()
memcache_debug()
memory_get_usage()
metaphone()
method_exists()
mhash()
mhash_count()
mhash_get_block_size()
mhash_get_hash_name()
mhash_keygen_s2k()
microtime()
mime_content_type()
min()
ming_setcubicthreshold()
ming_setscale()
ming_useswfversion()
mkdir()
mktime()
money_format()
move_uploaded_file()
msession_connect()
msession_count()
msession_create()
msession_destroy()
msession_disconnect()
msession_find()
msession_get()
msession_get_array()
msession_get_data()
msession_inc()
msession_list()
msession_listvar()
msession_lock()
msession_plugin()
msession_randstr()
msession_set()
msession_set_array()
msession_set_data()
msession_timeout()
msession_uniq()
msession_unlock()
msg_get_queue()
msg_receive()
msg_remove_queue()
msg_send()
msg_set_queue()
msg_stat_queue()
msql()
msql_affected_rows()
msql_close()
msql_connect()
msql_create_db()
msql_createdb()
msql_data_seek()

mysql_num_rows()
mysql_pconnect()
mysql_ping()
mysql_query()
mysql_real_escape_string()
mysql_result()
mysql_select_db()
mysql_stat()
mysql_tablename()
mysql_thread_id()
mysql_unbuffered_query()
mysqli()()
mysqli->affected_rows()
mysqli->autocommit()
mysqli->change_user()
mysqli->character_set_name()
mysqli->close()
mysqli->commit()
mysqli->disable_reads_from_master()
mysqli->dump_debug_info()
mysqli->errno()
mysqli->fetch_assoc()
mysqli->field_count()
mysqli->get_host_info()
mysqli->info()
mysqli->insert_id()
mysqli->kill()
mysqli->more_results()
mysqli->multi_query()
mysqli->next_result()
mysqli->options()
mysqli->ping()
mysqli->prepare()
mysqli->protocol_version()
mysqli->query()
mysqli->real_connect()
mysqli->real_escape_string()
mysqli->real_query()
mysqli->rollback()
mysqli->rpl_query_type()
mysqli->select_db()
mysqli->send_query()
mysqli->server_info()
mysqli->sqlstate()
mysqli->ssl_set()
mysqli->stat()
mysqli->stmt_init()
mysqli->store_result()
mysqli->thread_id()
mysqli->use_result()
mysqli->warning_count()
mysqli_affected_rows()
mysqli_autocommit()
mysqli_bind_param()

mysqli_bind_result()
mysqli_change_user()
mysqli_character_set_name()
mysqli_client_encoding()
mysqli_close()
mysqli_commit()
mysqli_connect()
mysqli_connect_errno()
mysqli_connect_error()
mysqli_data_seek()
mysqli_debug()
mysqli_disable_reads_from_master()
mysqli_disable_rpl_parse()
mysqli_dump_debug_info()
mysqli_embedded_connect()
mysqli_enable_reads_from_master()
mysqli_enable_rpl_parse()
mysqli_errno()
mysqli_error()
mysqli_escape_string()
mysqli_execute()
mysqli_fetch()
mysqli_fetch_array()
mysqli_fetch_assoc()
mysqli_fetch_field()
mysqli_fetch_field_direct()
mysqli_fetch_fields()
mysqli_fetch_lengths()
mysqli_fetch_object()
mysqli_fetch_row()
mysqli_field_count()
mysqli_field_seek()
mysqli_field_tell()
mysqli_free_result()
mysqli_get_client_info()
mysqli_get_client_version()
mysqli_get_host_info()
mysqli_get_metadata()
mysqli_get_proto_info()
mysqli_get_server_info()
mysqli_get_server_version()
mysqli_info()
mysqli_init()
mysqli_insert_id()
mysqli_kill()
mysqli_master_query()
mysqli_more_results()
mysqli_multi_query()
mysqli_next_result()
mysqli_num_fields()
mysqli_num_rows()
mysqli_options()
mysqli_param_count()
mysqli_ping()

N

natsort()
ncurses_addch()
ncurses_addchnstr()
ncurses_addchstr()
ncurses_addnstr()
ncurses_addstr()
ncurses_assume_default_colors()
ncurses_attroff()
ncurses_attron()
ncurses_attrset()
ncurses_baudrate()
ncurses_beep()
ncurses_bkgd()
ncurses_bkgdset()
ncurses_border()
ncurses_bottom_panel()
ncurses_can_change_color()
ncurses_cbreak()
ncurses_clear()
ncurses_clrtobot()
ncurses_clrtoeol()
ncurses_color_content()
ncurses_color_set()
ncurses_curs_set()
ncurses_def_prog_mode()
ncurses_def_shell_mode()
ncurses_define_key()
ncurses_del_panel()
ncurses_delay_output()
ncurses_delch()
ncurses_deleteln()
ncurses_delwin()
ncurses_doupdate()
ncurses_echo()
ncurses_echochar()
ncurses_end()
ncurses_erase()
ncurses_erasechar()
ncurses_filter()
ncurses_flash()
ncurses_flushinp()
ncurses_getch()
ncurses_getmaxyx()
ncurses_getmouse()
ncurses_getyx()
ncurses_halfdelay()
ncurses_has_colors()
ncurses_has_ic()
ncurses_has_il()
ncurses_has_key()
ncurses_hide_panel()
ncurses_hline()
ncurses_inch()
ncurses_init()

ncurses_init_color()
ncurses_init_pair()
ncurses_insch()
ncurses_insdelln()
ncurses_insertln()
ncurses_insstr()
ncurses_instr()
ncurses_isendwin()
ncurses_keyok()
ncurses_keypad()
ncurses_killchar()
ncurses_longname()
ncurses_meta()
ncurses_mouse_trafo()
ncurses_mouseinterval()
ncurses_mousemask()
ncurses_move()
ncurses_move_panel()
ncurses_mvaddch()
ncurses_mvaddchnstr()
ncurses_mvaddchstr()
ncurses_mvaddnstr()
ncurses_mvaddstr()
ncurses_mvcur()
ncurses_mvdelch()
ncurses_mvgetch()
ncurses_mvhline()
ncurses_mvinch()
ncurses_mvvline()
ncurses_mvwaddstr()
ncurses_napms()
ncurses_new_panel()
ncurses_newpad()
ncurses_newwin()
ncurses_nl()
ncurses_nocbreak()
ncurses_noecho()
ncurses_nonl()
ncurses_noqiflush()
ncurses_noraw()
ncurses_pair_content()
ncurses_panel_above()
ncurses_panel_below()
ncurses_panel_window()
ncurses_pnoutrefresh()
ncurses_prefresh()
ncurses_putp()
ncurses_qiflush()
ncurses_raw()
ncurses_refresh()
ncurses_replace_panel()
ncurses_reset_prog_mode()
ncurses_reset_shell_mode()
ncurses_resetty()

ncurses_savetty()
ncurses_scr_dump()
ncurses_scr_init()
ncurses_scr_restore()
ncurses_scr_set()
ncurses_scrl()
ncurses_show_panel()
ncurses_slk_attr()
ncurses_slk_attroff()
ncurses_slk_attron()
ncurses_slk_attrset()
ncurses_slk_clear()
ncurses_slk_color()
ncurses_slk_init()
ncurses_slk_noutrefresh()
ncurses_slk_refresh()
ncurses_slk_restore()
ncurses_slk_set()
ncurses_slk_touch()
ncurses_standend()
ncurses_standout()
ncurses_start_color()
ncurses_termattrs()
ncurses_termname()
ncurses_timeout()
ncurses_top_panel()
ncurses_typeahead()
ncurses_ungetch()
ncurses_ungetmouse()
ncurses_update_panels()
ncurses_use_default_colors()
ncurses_use_env()
ncurses_use_extended_names()
ncurses_vidattr()
ncurses_vline()
ncurses_waddch()
ncurses_waddstr()
ncurses_wattroff()
ncurses_wattron()
ncurses_wattrset()
ncurses_wborder()
ncurses_wclear()
ncurses_wcolor_set()
ncurses_werase()
ncurses_wgetch()
ncurses_whline()
ncurses_wmouse_trafo()
ncurses_wmove()
ncurses_wnoutrefresh()
ncurses_wrefresh()
ncurses_wstandend()
ncurses_wstandout()
ncurses_wvline()
next()

O

P

pdf_show_xy()
pdf_skew()
pdf_stringwidth()
pdf_stroke()
pdf_translate()
PDO::__construct()
PDO::beginTransaction()
PDO::commit()
PDO::errorCode()
PDO::errorInfo()
PDO::exec()
PDO::lastInsertId()
PDO::prepare()
PDO::rollBack()
PDO::setAttribute()
PDOStatement::bindColumn()
PDOStatement::bindParam()
PDOStatement::errorCode()
PDOStatement::errorInfo()
PDOStatement::execute()
PDOStatement::fetch()
PDOStatement::fetchAll()
PDOStatement::fetchSingle()
PDOStatement::rowCount()
pfpro_cleanup()
pfpro_init()
pfpro_process()
pfpro_process_raw()
pfpro_version()
pfsockopen()
pg_affected_rows()
pg_cancel_query()
pg_client_encoding()
pg_close()
pg_connect()
pg_connection_busy()
pg_connection_reset()
pg_connection_status()
pg_convert()
pg_copy_from()
pg_copy_to()
pg_dbname()
pg_delete()
pg_end_copy()
pg_escape_bytea()
pg_escape_string()
pg_fetch_all()
pg_fetch_array()
pg_fetch_assoc()
pg_fetch_object()
pg_fetch_result()
pg_fetch_row()
pg_field_is_null()
pg_field_name()

pg_field_num()
pg_field_prtlen()
pg_field_size()
pg_field_type()
pg_free_result()
pg_get_notify()
pg_get_pid()
pg_get_result()
pg_host()
pg_insert()
pg_last_error()
pg_last_notice()
pg_last_oid()
pg_lo_close()
pg_lo_create()
pg_lo_export()
pg_lo_import()
pg_lo_open()
pg_lo_read()
pg_lo_read_all()
pg_lo_seek()
pg_lo_tell()
pg_lo_unlink()
pg_lo_write()
pg_meta_data()
pg_num_fields()
pg_num_rows()
pg_options()
pg_parameter_status()
pg_pconnect()
pg_ping()
pg_port()
pg_put_line()
pg_query()
pg_result_error()
pg_result_seek()
pg_result_status()
pg_select()
pg_send_query()
pg_set_client_encoding()
pg_trace()
pg_tty()
pg_unescape_bytea()
pg_untrace()
pg_update()
pg_version()
php_check_syntax()
php_ini_scanned_files()
php_logo_guid()
php_register_url_stream_wrapper()
php_sapi_name()
php_stream_can_cast()
php_stream_cast()
php_stream_close()

php_stream_closedir()
php_stream_copy_to_mem()
php_stream_copy_to_stream()
php_stream_eof()
php_stream_filter_register_factory()
php_stream_filter_unregister_factory()
php_stream_flush()
php_stream_fopen_from_file()
php_stream_fopen_temporary_file()
php_stream_fopen_tmpfile()
php_stream_getc()
php_stream_gets()
php_stream_is()
php_stream_is_persistent()
php_stream_make_seekable()
php_stream_open_wrapper()
php_stream_open_wrapper_as_file()
php_stream_open_wrapper_ex()
php_stream_opendir()
php_stream_passthru()
php_stream_read()
php_stream_readdir()
php_stream_rewinddir()
php_stream_seek()
php_stream_sock_open_from_socket()
php_stream_sock_open_host()
php_stream_sock_open_unix()
php_stream_stat()
php_stream_stat_path()
php_stream_tell()
php_stream_write()
php_strip_whitespace()
php_uname()
php_unregister_url_stream_wrapper()
phpcredits()
phpinfo()
phpversion()
pi()
png2wbmp()
popen()
pos()
posix_ctermid()
posix_get_last_error()
posix_getcwd()
posix_getegid()
posix_geteuid()
posix_getgid()
posix_getgrgid()
posix_getgrnam()
posix_getgroups()
posix_getlogin()
posix_getpgid()
posix_getpgrp()
posix_getpid()

printer_select_brush()
printer_select_font()
printer_select_pen()
printer_set_option()
printer_start_doc()
printer_start_page()
printer_write()
printf()
proc_close()
proc_get_status()
proc_nice()
proc_open()
proc_terminate()
pspell_add_to_personal()
pspell_add_to_session()
pspell_check()
pspell_clear_session()
pspell_config_create()
pspell_config_data_dir()
pspell_config_dict_dir()
pspell_config_ignore()
pspell_config_mode()
pspell_config_personal()
pspell_config_repl()
pspell_config_runtogether()
pspell_config_save_repl()
pspell_new()
pspell_new_config()
pspell_new_personal()
pspell_save_wordlist()
pspell_store_replacement()
pspell_suggest()
putenv()

Q

qdom_error()
qdom_tree()
quoted_printable_decode()
quotemeta()

R

rad2deg()
rand()
range()
Rar::extract()
Rar::getAttr()
Rar::getCrc()
Rar::getFileTime()

Rar::getHostOs()
Rar::getMethod()
Rar::getName()
Rar::getPackedSize()
Rar::getUnpackedSize()
Rar::getVersion()
rar_close()
rar_entry_get()
rar_list()
rar_open()
rawurldecode()
rawurlencode()
read_exif_data()
readdir()
readfile()
readgzfile()
readline()
readline_add_history()
readline_callback_handler_install()
readline_callback_handler_remove()
readline_callback_read_char()
readline_clear_history()
readline_completion_function()
readline_info()
readline_list_history()
readline_on_new_line()
readline_read_history()
readline_redisplay()
readline_write_history()
readlink()
realpath()
recode()
recode_file()
recode_string()
RecursiveDirectoryIterator::getChildren()
RecursiveDirectoryIterator::hasChildren()
RecursiveDirectoryIterator::key()
RecursiveDirectoryIterator::next()
RecursiveDirectoryIterator::rewind()
RecursiveIteratorIterator::current()
RecursiveIteratorIterator::getDepth()
RecursiveIteratorIterator::getSubIterator()
RecursiveIteratorIterator::key()
RecursiveIteratorIterator::next()
RecursiveIteratorIterator::rewind()
RecursiveIteratorIterator::valid()
register_shutdown_function()
register_tick_function()
rename()
rename_function()
reset()
restore_error_handler()
restore_exception_handler()
restore_include_path()

S

session_id()
session_is_registered()
session_module_name()
session_name()
session_regenerate_id()
session_register()
session_save_path()
session_set_cookie_params()
session_set_save_handler()
session_start()
session_unregister()
session_unset()
session_write_close()
set_error_handler()
set_exception_handler()
set_file_buffer()
set_include_path()
set_magic_quotes_runtime()
set_time_limit()
setcookie()
setlocale()
setrawcookie()
settype()
sha1()
sha1_file()
shell_exec()
shm_attach()
shm_detach()
shm_get_var()
shm_put_var()
shm_remove()
shm_remove_var()
shmop_close()
shmop_delete()
shmop_open()
shmop_read()
shmop_size()
shmop_write()
show_source()
shuffle()
similar_text()
simplexml_import_dom()
simplexml_load_file()
simplexml_load_string()
SimpleXMLElement->asXML()
SimpleXMLElement->attributes()
SimpleXMLElement->children()
SimpleXMLElement->xpath()
SimpleXMLIterator::current()
SimpleXMLIterator::getChildren()
SimpleXMLIterator::hasChildren()
SimpleXMLIterator::key()
SimpleXMLIterator::next()
SimpleXMLIterator::rewind()

SimpleXMLIterator::valid()
sin()
sinh()
sizeof()
sleep()
snmp_get_quick_print()
snmp_get_valueretrieval()
snmp_read_mib()
snmp_set_enum_print()
snmp_set_oid_numeric_print()
snmp_set_quick_print()
snmp_set_valueretrieval()
snmpget()
snmpgetnext()
snmprealwalk()
snmpset()
snmpwalk()
snmpwalkoid()
SoapClient::__call()
SoapClient::__getFunctions()
SoapClient::__getLastRequest()
SoapClient::__getLastResponse()
SoapClient::__getTypes()
SoapClient::SoapClient()
SoapFault::SoapFault()
SoapHeader::SoapHeader()
SoapParam::SoapParam()
SoapServer::addFunction()
SoapServer::getFunctions()
SoapServer::handle()
SoapServer::setClass()
SoapServer::setPersistence()
SoapServer::SoapServer()
SoapVar::SoapVar()
socket_accept()
socket_bind()
socket_clear_error()
socket_close()
socket_connect()
socket_create()
socket_create_listen()
socket_create_pair()
socket_get_option()
socket_get_status()
socket_getpeername()
socket_getsockname()
socket_last_error()
socket_listen()
socket_read()
socket_recv()
socket_recvfrom()
socket_select()
socket_send()
socket_sendto()

srand()
sscanf()
ssh2_auth_none()
ssh2_auth_password()
ssh2_auth_pubkey_file()
ssh2_connect()
ssh2_exec()
ssh2_fetch_stream()
ssh2_fingerprint()
ssh2_methods_negotiated()
ssh2_scp_recv()
ssh2_scp_send()
ssh2_sftp()
ssh2_sftp_lstat()
ssh2_sftp_mkdir()
ssh2_sftp_readlink()
ssh2_sftp_realpath()
ssh2_sftp_rename()
ssh2_sftp_rmdir()
ssh2_sftp_stat()
ssh2_sftp_symlink()
ssh2_sftp_unlink()
ssh2_shell()
ssh2_tunnel()
stat()
stmt->bind_param()
stmt->bind_result()
stmt->data_seek()
stmt->execute()
stmt->fetch()
stmt->free_result()
stmt->num_rows()
stmt->param_count()
stmt->prepare()
stmt->reset()
stmt->send_long_data()
str_ireplace()
str_pad()
str_repeat()
str_replace()
str_rot13()
str_shuffle()
str_split()
str_word_count()
strcasecmp()
strchr()
strcmp()
strcoll()
strcspn()
stream_context_create()
stream_context_get_default()
stream_context_get_options()
stream_context_set_option()
stream_context_set_params()

stream_copy_to_stream()
stream_filter_append()
stream_filter_prepend()
stream_filter_register()
stream_filter_remove()
stream_get_contents()
stream_get_filters()
stream_get_line()
stream_get_meta_data()
stream_get_transports()
stream_get_wrappers()
stream_register_wrapper()
stream_select()
stream_set_blocking()
stream_set_timeout()
stream_set_write_buffer()
stream_socket_accept()
stream_socket_client()
stream_socket_enable_crypto()
stream_socket_get_name()
stream_socket_pair()
stream_socket_recvfrom()
stream_socket_sendto()
stream_socket_server()
stream_wrapper_register()
stream_wrapper_restore()
stream_wrapper_unregister()
strftime()
strip_tags()
stripcslashes()
stripos()
stripslashes()
stristr()
strlen()
strnatcasecmp()
strnatcmp()
strncasecmp()
strncmp()
strpbrk()
strpos()
strptime()
strrchr()
strrev()
strripos()
strrpos()
strspn()
strstr()
strtok()
strtolower()
strtotime()
strtoupper()
strtr()
strval()
substr()

swf_shapecurveto3()
swf_shapefillbitmapclip()
swf_shapefillbitmaptile()
swf_shapefilloff()
swf_shapefillsolid()
swf_shapelinesolid()
swf_shapelineto()
swf_shapemoveto()
swf_showframe()
swf_startbutton()
swf_startdoaction()
swf_startshape()
swf_startsymbol()
swf_textwidth()
swf_translate()
swf_viewport()
swfaction()
swfbitmap()
SWFBitmap->getHeight()
SWFBitmap->getWidth()
swfbutton()
SWFbutton->addAction()
SWFbutton->addShape()
SWFbutton->setAction()
SWFbutton->setdown()
SWFbutton->setHit()
SWFbutton->setOver()
SWFbutton->setUp()
swfbutton_keypress()
swfdisplayitem()
SWFDisplayItem->addColor()
SWFDisplayItem->move()
SWFDisplayItem->moveTo()
SWFDisplayItem->multColor()
SWFDisplayItem->remove()
SWFDisplayItem->Rotate()
SWFDisplayItem->rotateTo()
SWFDisplayItem->scale()
SWFDisplayItem->scaleTo()
SWFDisplayItem->setDepth()
SWFDisplayItem->setName()
SWFDisplayItem->setRatio()
SWFDisplayItem->skewX()
SWFDisplayItem->skewXTo()
SWFDisplayItem->skewY()
SWFDisplayItem->skewYTo()
swffill()
SWFFill->moveTo()
SWFFill->rotateTo()
SWFFill->scaleTo()
SWFFill->skewXTo()
SWFFill->skewYTo()
swffont()
swffont->getwidth()

swfgradient()
SWFGradient->addEntry()
swfmorph()
SWFMorph->getshape1()
SWFMorph->getshape2()
swfmovie()
SWFMovie->add()
SWFMovie->nextframe()
SWFMovie->output()
swfmovie->remove()
SWFMovie->save()
SWFMovie->setbackground()
SWFMovie->setdimension()
SWFMovie->setframes()
SWFMovie->setrate()
SWFMovie->streammp3()
swfshape()
SWFShape->addFill()
SWFShape->drawCurve()
SWFShape->drawCurveTo()
SWFShape->drawLine()
SWFShape->drawLineTo()
SWFShape->movePen()
SWFShape->movePenTo()
SWFShape->setLeftFill()
SWFShape->setLine()
SWFShape->setRightFill()
swfsprite()
swfsprite->add()
SWFSprite->nextframe()
SWFSprite->remove()
SWFSprite->setframes()
swftext()
SWFText->addString()
SWFText->getWidth()
SWFText->moveTo()
SWFText->setColor()
SWFText->setFont()
SWFText->setHeight()
SWFText->setSpacing()
swftextfield()
SWFTextField->addstring()
SWFTextField->align()
SWFTextField->setbounds()
SWFTextField->setcolor()
SWFTextField->setFont()
SWFTextField->setHeight()
SWFTextField->setindentation()
SWFTextField->setLeftMargin()
SWFTextField->setLineSpacing()
SWFTextField->setMargins()
SWFTextField->setname()
SWFTextField->setrightMargin()
sybase_affected_rows()

T

tidy_is_xhtml()
tidy_is_xml()
tidy_load_config()
tidy_node->children()
tidy_node->get_attr()
tidy_node->get_nodes()
tidy_node->hasChildren()
tidy_node->hasSiblings()
tidy_node->isComment()
tidy_node->isHtml()
tidy_node->isJste()
tidy_node->isText()
tidy_node->isXhtml()
tidy_node->isXml()
tidy_node->next()
tidy_node->prev()
tidy_parse_file()
tidy_parse_string()
tidy_repair_file()
tidy_repair_string()
tidy_reset_config()
tidy_save_config()
tidy_set_encoding()
tidy_setopt()
tidy_warning_count()
tidyNode->isAsp()
tidyNode->isPhp()
time()
time_nanosleep()
tmpfile()
token_get_all()
token_name()
touch()
trigger_error()
trim()

U

uasort()
ucfirst()
ucwords()
udm_add_search_limit()
udm_alloc_agent()
udm_alloc_agent_array()
udm_api_version()
udm_cat_list()
udm_cat_path()
udm_check_charset()
udm_check_stored()
udm_clear_search_limits()
udm_close_stored()
udm_crc32()

V

W

X

Y

Z

Apéndice T. Material que falta