Capítulo 15: Manejo de HTTP en PHP.

Uno de los temas más importantes en todo lenguaje de scripting usado para generación dinámica de sitios web y aplicaciones web es el manejo del protocolo HTTP, conexiones, métodos GET y POST, uploads, headers, cache y demás alternativas. Todas estas funciones están bien soportadas en php de forma tal de tener desde el lneguaje un control completo sobre la forma en que el server interactua con el browser.

Headers.

Una de las funciones más importantes de PHP es la función "header" que sirve para enviar al browser un determinado header HTTP, por default en cuanto un script PHP usa una función de salida o transmite algo al browser php envía el header "Content-Type: text/html" al browser. Por eso es importante destacar que la funcion header solo puede usarse ANTES de realizar cualquier tipo de salida al browser.

Ejemplos:

```
header("Location: http://lugar/pepe.php");
```

Este es un header http que sirve para redireccionar al browser a otra página, script o URL, es muy util para scripts php que procesan datos recibidos desde un formulario o similar y luego en funcion de los datos redireccionan al browser a una página que por ejemplo puede mostrar errores en el ingreso de datos, aceptar los datos o simplemente volver a la página que llamo al script.

Este tipo de redirección sólo puede usarse si no se emitió ninguna salida al browser, si ya se emitió una salida al browser y es necesario redireccionar la página debemos generar desde PHP código JavaScript que transmitimos al browser y el browser si es capaz de interpretar JavaScript podrá redireccionarse a la página pasada, por ejemplo:

```
<?
print("Cosas anteriores");
if($redireccionar) {
 ?>
  <script>
  location.href=<?print($URL);?>
  </script>
 ?<
}
?>
```

El browser recibe por ejemplo:

```
<script>
location..href=http://www.yahoo.com
</script>
```

Y automaticamente se redirecciona al URL pasado, el uso de JavaScript desde PHP es importante (ver el capítulo de programacion en el cliente desde el server)

Otro header importante que ya vimos es el de imagen que enviamos antes de pasarle al browser una imagen generada dinámicamente.

Una de las funciones importantes de los headers HTPP es controlar el comportamiento del cache del browser, en numerosas ocasiones tendremos que codificar scripts que generan una cierta salida que no es deseable que sea cacheable, si no impedimos que el browser cachee el resultado del script el usuario tendrá que refrescar manualmente la página desde el browser para ver el nuevo resultado. Los siguientes headers http impiden que el browser cachee una página y sirven tanto para Internet Explorer como para Netscape Navigator:

```
header("Cache-Control: no-cache, must-revalidate");
header("Pragma: no-cache");
header("Expires: Mon,26 Jul 1997 05:00:00 GMT");
```

El protocolo HTTP dispone de un mecanismo de autorización básico mediante el cual puede pedirse al browser que promptee al usuario por un usuario y password, luego estos datos se transmiten al web-server y el mismo debe encargarse de autorizar al usuario o no según corresponda. Desde PHP podemos generar los headers correspondientes para que el browser pida al usuario un usuario y password y luego validar estos datos desde el script php (usando una base de datos o lo que corresponda):

```
<?php
if(!isset($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\n";
exit;} else {
echo "Hello $PHP_AUTH_USER.<P>";
echo "You entered $PHP_AUTH_PW as your password.<P>";
}
?>
```

El header “WWW-Authenticate: Basic realm=\”My Realm\”” fuerza al browser a preguntar al usuario por un user y password, si el usuario cancela el script sigue su ejecución en la próxima línea (notar que enviamos un header de acceso no autorizado y termina el script”). Si el usuario ingresa un usuario y password el browser vuelve a invocar al script pasándole como variables $PHP_AUTH_USER y $PHP_AUTH_PW.

Control de la conexión usando PHP.

Internamente PHP mantiene un status de la conexión con el cliente durante la ejecución del script, el valor del status es habitualmente “NORMAL”, pero puede cambiar a “ABORTED” si el usuario presiona el botón de STOP o a “TIEMOUT” si es script sobrepasa el tiempo limite de ejecución que se fija en (/var/lib/php.ini). Es posible controlar que se hace si el usuario interrumpe la ejecución de un script con el botón de STOP usando dos funciones de PHP:

boolean=connection_aborted()

Devuelve true si la conexión fue abortada por el usuario presionando STOP, cerrando el browser o similar.

register_shutdown_function(nombre_funcion)

Esta funcion permite registrar el nombre de una función que se ejecutara en los siguientes casos:

a) Cuando el script intenta enviar algo al browser pero el usuario aborto la conexión.
b) Cuando termina normalmente la ejecución del script.

Para distinguir estos dos casos dentro de la función que se registre es importante usar la función connection_aborted para saber si la función se llama por la terminación normal del script o por conexión abortada. La función registrada con register_shutdown_function no puede generar ninguna salida (lo cual dificulta debuggearla), usualmente sirve para chequear consistencia en bases de datos, eliminar archivos temporales que pueden quedar abiertos, loggear un problema, registrar que la conexión fue abortada, hacer rollback de una transacción o similares.

Transacciones HTTP usando PHP.

Una modalidad muy en boga en estos días es el establecimiento de servidores de transacciones HTTP o HTTPS (usando SSL), muchas veces el propósito de estos servidores es hacer de gateway para solicitar servicios a una determinada red de servidores. Por ejemplo supongamos que el site www.super.com tiene 56 servidores, para poder interactuar con empresas externas www.super.com muchas veces tiene el problema de que un agente externo debe desencadenar una acción en uno de sus web servers, setear "n" servers para permitir que algunas direcciones en particular tengan acceso a cierta funcionalidad es muy complejo por lo que www.super.com decide instalar un servidor de transacciones, cualquiera tiene acceso a este servidor para solicitar servicios usando el protocolo HTTP y este servidor tiene acceso a los servers de www.super.com, el unico cambio en los 56 servers es permitir que el servidor de transacciones opere sobre ellos.

Para generar transacciones HTTP muchas veces es necesario que un scrip "simule" postear datos a un script como si los datos vinieran de un formulario HTML, a continuación sugerimos 2 distintas formas de generar transacciones POST desde un script sin necesidad de que un usuario submitee los datos:

Método 1: Usando CURL.

Curl es una pequeña utilidad que corre en Unix o Windows y permite generar paquetes HTTP de todo tipo, desde PHP podemos llamar a CURL como un programa externo para generar la transacción de la forma:

```
<?php
$data="hola=mundo";
$URL="luigi.arcadia.com.ar/pruebas/profo.php";
exec("/usr/local/bin/curl -m 120 -d \"$data\" http://$URL -L",$return_msg_array,$return_number);
for($i=0;$i<count($return_msg_array);$i++) {
  $results=$results.$return_msg_array[$i];
}
print("Resultado:$results\n");
```

Como vemos "curl" es llamado usando la función "exec" al ser un programa externo, $data tiene los datos a enviar en formato "nombre1=valor1&nombre2=valor2....", curl devuelve un vector donde cada elemento del vector es una línea de la salida que devuelve el server al recibir el request en modo POST.
En este script como vemos pegamos todas las líneas de salida y mostramos el resultado en pantalla.

Método 2: Usando la clase http_post.

Si no que quiere usar un programa externa podemos usar una clase que denominamos http_post y permite postear datos en un servidor, un ejemplo de uso de esta clase es:

```
<?
include("class_post.php");
$a=new http_post;
$a->set_action("http://luigi.arcadia.com.ar/pruebas/profo.php");
$a->set_element("hola","mundo");
$res=$a->send(0);
print("Resultado: $res");
?>
```

Como vemos en este script hacemos lo mismo que en el anterior, la diferencia principal radica en que esta clase devuelve como resultado "TODA" la salida del web server (incluyendo headers) mientras que CURL en la forma en que fue llamado no devuelve los headers.

La clase http_post es la siguiente:

```php
<?php
#
# http_post - PHP3 class for posting a 'form' from within a php3 script
# Version 0.5b
#
# Copyright 2000
# Alan van den Bosch (alan@sanguis.com.au)
# Sanguis Pty Ltd (acn 061 444 031)
#
# Licence:
# You are granted the right to use and/or redistribute this
# code only if this licence and the copyright notice are included
# and you accept that no warranty of any kind is made or implied
# by the author or Sanguis Pty Ltd.
#
#
# Methods:
#
# http_post()
# Constructor used when creating a new instance of the http_post class.
# Returns true on success.
# ie.
# $a=new http_post;
#
#
# set_server(string SERVER)
# Set the server of the URI you wish to post to. see also set_action()
# Returns true on success.
# ie.
# $a->set_server("127.0.0.1");
# or
# $a->set_server("www.somehost.org");
#
#
# set_port(string PORT)
# Set the tcp port of the URI you wish to post to. see also set_action()
# Returns true on success.
# ie.
# $a->set_port("8080");
#
#
# set_file(string FILENAME)
# Set the filename of the URI you wish to post to. see also set_action()
# Returns true on success.
# ie.
# $a->set_file("/incoming.php3");
#
#
# set_action(string ACTION)
# Set the URI you wish to post to.
# Returns true on success.
# ie.
# $a->set_action("http://www.somehost.org:8080/incoming.php3");
#
# set_enctype(string ENCTYPE)
# Set the encoding type used for the post. Can have the values
# "application/x-www-form-urlencoded" or "multipart/form-data"
```

```
# Returns true on success.
# ie.
# $a->set_enctype("multipart/form-data");
#
#
# set_element(string NAME, string VALUE)
# Set or update a single name/value pair to be posted
# Returns true on success.
# ie.
# $a->set_element("username","John Doe");
#
#
# set_element(array ELEMENTS)
# Set or update a number of name/value pairs to be posted
# Returns true on success.
# ie.
# $a->set_element(array("username" => "John Doe",
# "password" => "dead-ringer",
# "age" => "99"));
#
#
# set_timeout(integer TIMEOUT)
# Set the number of seconds to wait for the server to connect
# when posting. minimum value of 1 second.
# Returns true on success.
# ie.
# $a->set_timeout(10);
#
# show_post()
# Show the current internal state of an instance, for debugging.
# Returns true on success.
# ie.
# $a->show_post();
#
#
# send(boolean DISPLAY)
# Send the name/value pairs using the post method. The response
# can be echoed by setting DISPLAY to a true value.
# Returns a string containing the raw response on success, false
# on failure.
# ie.
# $a->send(1);
#


class http_post
{
function http_post(){
$this->_method= "post";
$this->_server=$GLOBALS[ "HTTP_HOST"];
$this->_file= "\\";
$this->_port= "80";
$this->_enctype= "application/x-www-form-urlencoded";
$this->_element=array();
$this->_timeout=20;
}
```

```
function set_server($newServer= ""){
if(strlen($newServer)<1)$newServer=$HTTP_HOST;
$this->_server=$newServer;
return 1;
}

function set_port($newPort= "80"){
$newPort=intval($newPort);
if($newPort < 0 || $newPort > 65535)$newPort=80;
$this->_port=$newPort;
return 1;
}

function set_file($newFile= "\\"){
$this->_file=$newFile;
return 1;
}

function set_action($newAction= ""){
$pat= "^((http://){1}([^:/]{0,}){1}(:([0-9]{1,})){0,1}){0,1}(.*)";

if(eregi($pat,$newAction,$sub)){
if(strlen($sub[3])>0)$this->_server=$sub[3];
if(strlen($sub[5])>0)$this->_port=$sub[5];
$this->_file=$sub[6];
return 1;
}
return 0;
}

function set_enctype($newEnctype= "application/x-www-form-urlencoded"){
if($newEnctype != "application/x-www-form-urlencoded" &&
$newEnctype != "multipart/form-data"){
$newEnctype= "application/x-www-form-urlencoded";
}
$this->_enctype=$newEnctype;
return 1;
}

function set_element($key= "",$val= ""){
if(is_array($key)){
$len=sizeof($key);
reset($key);
for($i=0;$i<$len;$i++){
$cur=each($key);
$k=$cur[ "key"];
$v=$cur[ "value"];
$this->_element[$k]=$v;
}
}
else{
if(strlen($key)>0)$this->_element[$key]=$val;
}
return 1;
}

function set_timeout($newTimeout=20){
```

```
$newTimeout=intval($newTimeout);
if($newTimeout<1)$newTimeout=1;
$this->_timeout=$newTimeout;
return 1;
}

function show_post(){
$str= "";
$str.= "Action:".$this->_action. "<br>";
$str.= "Server:".$this->_server. "<br>";
$str.= "Port:".$this->_port. "<br>";
$str.= "File:".$this->_file. "<br>";
$str.= "Enctype:".$this->_enctype. "<br>";

echo $str;

$len=sizeof($this->_element);
reset($this->_element);
for($i=0;$i<$len;$i++){
$cur=each($this->_element);
$key=$cur[ "key"];
$val=$cur[ "value"];
echo "Field:$key = $val<br>\n";
}
return 1;
}

function send($display=0){
// open socket to server
$errno=$errstr=$retstr= "";
$sk = fsockopen($this->_server,
$this->_port,
&$errno,
&$errstr,
$this->_timeout
);
if(!$sk){
return 0;
}
else{
$boundary= "----".md5(uniqid(rand())). "----";
$message=$this->_get_message($boundary);
$str= "";
$str.=strtoupper($this->_method). " ";
$str.=$this->_file. " HTTP/1.0 \r\n";
$str.= "Referer: \r\n";
$str.= "User-Agent: php-HTTP_POST/1.0 \r\n";
$str.= "Host: ".$this->_server. "\r\n";

$str.= "Content-type: ".$this->_enctype;
if($this->_enctype== "multipart/form-data"){
$str.= "; boundary=".$boundary;
}
$str.= " \r\n";

$str.= "Content-length: ".strlen($message). "\r\n\r\n";
$str.=$message;
```

```
fputs($sk,$str);

while(!feof($sk)){
$resp=fgets($sk,80);
$retstr.=$resp;
if($display)echo $resp;
}

fclose($sk);
return $retstr;
}
}

function _get_message($boundary= ""){
$retstr= "";

$len=sizeof($this->_element);
reset($this->_element);

$switch=($this->_enctype== "multipart/form-data")?0:1;

for($i=0;$i<$len;$i++){
$cur=each($this->_element);
$key=$cur[ "key"];
$val=$cur[ "value"];

if($switch){
if(strlen($retstr)!=0)$retstr.= "&";
$retstr.=rawurlencode($key). "=";
$retstr.=rawurlencode($val);
}
else{
$retstr.=$boundary. "\r\n";
$retstr.= "Content-Disposition: form-data; ";
$retstr.= "name=\"$key\"\r\n\r\n$val\r\n\r\n";
}
}
if(!$switch)$retstr.=$boundary. "\r\n";
return $retstr;
}
}

?>
```

La clase utiliza las funciones de networking de php para conectarse al servidor indicado y enviar usando método post los datos que se desean.