Capítulo 9: Persistencia.

Uno de los problemas clásicos en el desarrollo de web sites y aplicaciones web es la perdida de persistencia cuando el usuario pasa de una página a otra. Debido a las características de diseño del protocolo HTTP que fuerza una nueva conexión y desconexión por cada request no es posible saber quien está accediendo a que página o en que lugar esta cada usuario del site. Mantener persistencia a lo largo de la navegación del sitio ha sido uno de los temas más complejos e importantes en el desarrollo de aplicaciones web, en este capítulo ilustraremos distintos métodos que pueden usarse en PHP para mantener persistencia.

Sesiones:

Se suele definir como una sesión al tiempo en el que un usuario determinado se encuentra navegando en el site, dependiendo de la definición podemos decir que si el usuario no navega por el site durante una cierta cantidad de minutos ha terminado su sesión en el sitio y a partir de allí cuando vuelve a ingresar lo hace en una nueva sesión. El concepto de sesión es útil porque es posible asociar a cada sesión un identificador único de forma tal de registrar la actividad del usuario en el site y mantener persistencia utilizando únicamente este identificador, el problema pasa a ser como mantener la persistencia del identificador de sesión (SID) de ahora en adelante, y las posibilidades son las que detallamos a continuación:

1. Cookies

Uno de los mecanismos más usados para mantener persistencia es el mecanismo de cookies, inventado por Netscape y hoy en día aceptado por casi todos los browsers, en especial los más populares. El concepto es que mediante un header del protocolo HTTP el server pueda almacenar información en el cliente. A esta información que el server guarda en el cliente se la denomina “cookie”. Las cookies pueden habilitarse o deshabilitarse desde el browser por lo que algunos usuarios no lo soportan, son de uso bastante general en muchos web sites a punto tal que en sites de la importancia de yahoo si el usuario no tiene habilitadas las cookies prácticamente no puede utilizar la mayoría de los servicios del site. Cuando el server envía un header con un cookie el browser, si acepta cookies, guarda la información enviada en un archivo de texto con un formato especial. Cada vez que el browser solicita una página del dominio que envió la cookie re-envia la cookie al site, de esta forma es posible mantener persistencia. La información que puede guardarse en una cookie esta limitada por lo que habitualmente se utiliza la misma para mantener el identificador de sesión del usuario almacenándose el resto de los datos necesarios en el servidor usando el session-id de la cookie como clave.

Para crear un cookie en PHP se utiliza la función setcookie cuya sintaxis es la siguiente:

int=setcookie(nombre, valor, expiración, path, dominio);

Nombre	: Nombre de la cookie a setear por ejemplo “sesion”
Valor	: Valor que contendrá la cookie, como por ejemplo “khdhkfdh47”
Expiracion	: Fecha de vencimiento de la cookie (fecha en la cual el browser la borra del disco del usuario), debe estar en formato Unix. En general el uso más practico es time()+tiempo donde tiempo es la cantidad de segundos de vida de la cookie.
Path	: En general no se usa, suele setearse en “/”
Dominio	: Dominio para el cual el cookie es valido ejemplo “.prueba.com” en cuyo caso sirve para algo.prueba.com, site1.prueba.com, site2.prueba.com y todos los de la misma forma.

La función devuelve verdadero si pudo setearse la cookie o falso en caso contrario (por ejemplo si el browser no acepta cookies)

Ejemplo:

$val=setcookie(“sesion”,”1”,time()+3600,”/”,”.prueba.com”);

Para recuperar el valor de una cookie se debe usar el vector de PHP $HTTP_COOKIE_VARS que es u vector asociativo indexado por nombre de cookie.

Ejemplo:

$ck=$HTTP_COOKIE_VARS[“sesion”];

Con lo cual se recupera el valor de la cookie.

Las cookies son sumamente practicas para manejar sesiones, en cada página se verifica si el usuario tiene seteada la cookie con nombre “session” si la tiene se recupera el valor de la sesión, si no la tiene se crea un identificador de sesión nuevo y único y se setea la cookie correspondiente con el vencimiento que corresponde según la duración que uno considere necesaria. Luego el identificador de sesión es accesible en cada página para almacenar valores por ejemplo en la base de datos como el nombre del usuario, preferencias de la sesión y otros valores. La recuperación de la cookie y su creación en caso de no existir se puede colocar en un modulo php que luego se incluye en cada página por ejemplo mod_session.php solo hay que recordar en cada página hacer un include(“mod_session.php”); y luego se puede usar la variable $session_id (o el nombre que se le haya dado en el modulo) para guardar y recuperar valores correspondientes a la sesión actual.

2. URL

Otro método posible para pasar información desde una página a otra es mediante el URL de la página en cuestión, usando el URL se pueden pasar datos de una página a otra usando el query string de la forma:

http://dominio/path?query_string

Donde query_string es de la forma: variable=valor&variable2=valor2&variable3=valor3 … etc…

De esta forma podríamos hacer un manejo similar al anterior pero pasando el session_id usando el url en lugar de usando cookies, la desventaja de este método es que todos los links deben generarse dinámicamente en PHP para agregar a la dirección del link el valor del cookie de la forma:

<a href=”http://dominio/path?session=<?print(“$session_id”);?>”>

El funcionamiento es similar, no requiere que el browser tenga habilitados cookies pero altera la forma en que se escriben los links y afea un poco la forma en la cual se muestra la URL de la página actual, podría por ejemplo tener consecuencias como entorpecer la tarea de generar el bookmark de una determinada página.

3. Sesiones en PHP

PHP soporta en forma nativa desde el lenguaje el concepto de sesiones, en PHP se pueden manejar sesiones en forma transparente al usuario, el lenguaje define una constante PHPSESSID con el identificador de cada sesión y se encarga de propagar el mismo usando cookies o bien el URL de la página en caso de que los cookies estén deshabilitados.

Para usar cookies en PHP es necesario invocar la función session_start(); en el comienzo del script, esta función se encarga de recuperar el session_id en la constante $PHPSESSID y si la sesión no existe crea una nueva.

Si PHP esta compilado con la opción –enable-trans-sid el interprete se encarga de re-escribir dinámicamente las URLs agregando el session_id en caso de que el usuario no tenga cookies, en caso contrario hay que modificar los links de la forma:

<a href=”/lugar?PHPSESSID=<?echo “$PHPSESSID”?>”>

PHP dispone además de funciones para “registrar” variables dentro de una sesión, esto se hace llamando a la función session_register(); Por ejemplo:

```
<?
session_start();
$x=”hola”;
session_register(“x”);
?>
```

Notar que session_register recibe el “nombre” de la variable a registrar sin “$”. Una vez registrada una variable la misma estará disponible en todas las páginas que usen session_start durante la sesión actual, es decir que si desde la página donde hicimos el session_register el usuario se traslada a otra página que usa session_start entonces la variable $x automáticamente estará definida y con el valor “hola” ya que es una variable registrada dentro de la sesión. Cualquier tipo de variable de PHP puede registrarse con este método, incluyendo objetos (sólo las propiedades del objeto se registran, no los métodos!)

Otras funciones de manejo de sesiones:

Función	Descripción
session_id();	Devuelve el id de la sesión actual, si se le pasa como parámetro un string cambia el id de la sesión al string pasado, esta función hace lo mismo que la consulta directa a la variable $PHPSESSID
session_destroy();	Destruye la sesión actual y todas las variables registradas.
session_name();	Devuelve el nombre de la sesión actual (PHPSESSID), si se le pasa como parámetro un string cambia el nombre de la sesión y por consiguiente de la variable que contiene el id de la sesión al valor pasado.
session_unregister(variable_name);	Elimina de la sesión una variable registrada con session_register
session_is_registered(variable_name);	Dice si una variable esta o no registrada en la sesión actual (devuelve true/false), notar que es equivalente a la función isset pero tomando en cuenta la sesión.

Manejo avanzado de sesiones en PHP.

Una pregunta interesante relativa al manejo de sesiones en PHP es como se almacenan los datos de la sesión (las variables registradas y demás), en PHP por omisión los datos sobre las sesiones se almacenan en archivos en el directorio /tmp, este método funciona correctamente pero tiene 2 inconvenientes:

1. No es fácil compartir sesiones entre servers.

   Dado que los datos se almacenan en un file-system una variable “registrada” en un server no puede ser visible en otro ya que el mismo no tiene forma de recuperar el contenido de la misma. (Podría hacerse si se cambia el directorio /tmp a un directorio compartido entre maquinas usando NFS pero este no es el tema aquí)

2. Dada una cantidad de usuarios realmente elevada el método no es eficiente.

Estamos hablando de cantidades de usuarios realmente muy grandes simultáneos, dados los cuales hay demasiadas sesiones por mantener y el file-system se torna lento en la búsqueda y recuperación de los archivos con los datos.

Afortunadamente en PHP es posible redefinir los métodos de almacenamiento que el lenguaje utiliza para almacenar las sesiones. Esto se hace con la función session_set_save_handler

Ejemplo:
session_set_save_handler ("open", "close", "read", "write", "destroy", "gc");

Que recibe como parámetros los "nombres" de 6(seis) funciones, cuyos prototipos y funcionalidad son los siguientes:

Función	Descripción
function open ($save_path, $session_name)	Se llama luego de cada session_start, recibe un path y un nombre de sesión que por omisión es PHPSESSID.
function close()	Cierra la sesión.
function read ($key)	Dada una clave "key" recupera para la sesión actual el valor de la clave pasada.
function write ($key, $val)	Guarda un par clave-valor para la sesión actual.
function destroy ($key)	Destruye la clave pasada como parámetro.
Function gc($maxlifetime)	Garbage collection, se encarga de destruir los datos de las sesiones vencidas.

En general en open no es necesario hacer nada ya que todo el manejo de Registración de variables esta dado por read y write. Write recibirá como clave el identificador de la sesión y como datos una estructura interna de php que contiene la representación de todos los datos a almacenar para la sesión. De la misma forma read debe ,dada la clave (session_id), recuperar todos los datos de la sesión y luego php en forma interna setea las variables correspondientes para que sean visibles.

Todas las funciones deben devolver "true" para que el manejo de sesiones funcione.

A continuación dos módulos de Zing Yiang ilustrando como hacer un manejador de sesiones usando DBM y otro usando MySQL y un ejemplo de test.

```
<?
/* ------------------------------------------------------------------------
 * session_dbm.php
 * ------------------------------------------------------------------------
 * PHP4 DBM Session Handler
 * Version 1.00
 * by Ying Zhang (ying@zippydesign.com)
 * Last Modified: May 21 2000
 *
 * ------------------------------------------------------------------------
 * TERMS OF USAGE:
 * ------------------------------------------------------------------------
 * You are free to use this library in any way you want, no warranties are
 * expressed or implied.  This works for me, but I don't guarantee that it
 * works for you, USE AT YOUR OWN RISK.
 *
 * While not required to do so, I would appreciate it if you would retain
 * this header information.  If you make any modifications or improvements,
 * please send them via email to Ying Zhang <ying@zippydesign.com>.
 *
 * ------------------------------------------------------------------------
 * DESCRIPTION:
```

```
 * ------------------------------------------------------------------------
 * This library tells the PHP4 session handler to write to a DBM file
 * instead of creating individual files for each session.
 *
 * ------------------------------------------------------------------------
 * INSTALLATION:
 * ------------------------------------------------------------------------
 * Make sure you have DBM support compiled into PHP4.  Then copy this
 * script to a directory that is accessible by the rest of your PHP
 * scripts.
 *
 * ------------------------------------------------------------------------
 * USAGE:
 * ------------------------------------------------------------------------
 * Include this file in your scripts before you call session_start(), you
 * don't have to do anything special after that.
 */

$SESS_DBM = "";
$SESS_LIFE = get_cfg_var("session.gc_maxlifetime");

function sess_open($save_path, $session_name) {
       global $SESS_DBM;

       $SESS_DBM = dbmopen("$save_path/$session_name", "c");
       return ($SESS_DBM);
}

function sess_close() {
       global $SESS_DBM;

       dbmclose($SESS_DBM);
       return true;
}

function sess_read($key) {
       global $SESS_DBM, $SESS_LIFE;

       $var = "";
       if ($tmp = dbmfetch($SESS_DBM, $key)) {
              $expires_at = substr($tmp, 0, strpos($tmp, "|"));

              if ($expires_at > time()) {
                     $var = substr($tmp, strpos($tmp, "|") + 1);
              }
       }

       return $var;
}

function sess_write($key, $val) {
       global $SESS_DBM, $SESS_LIFE;

       dbmreplace($SESS_DBM, $key, time() + $SESS_LIFE . "|" . $val);
       return true;
}

function sess_destroy($key) {
       global $SESS_DBM;

       dbmdelete($SESS_DBM, $key);
       return true;
}
```

```
function sess_gc($maxlifetime) {
	global $SESS_DBM;

	$now = time();
	$key = dbmfirstkey($SESS_DBM);
	while ($key) {
		if ($tmp = dbmfetch($SESS_DBM, $key)) {
			$expires_at = substr($tmp, 0, strpos($tmp, "|"));
			if ($now > $expires_at) {
				sess_destroy($key);
			}
		}

		$key = dbmnextkey($SESS_DBM, $key);
	}
}

session_set_save_handler(
	"sess_open",
	"sess_close",
	"sess_read",
	"sess_write",
	"sess_destroy",
	"sess_gc");
?>


<?
/* ------------------------------------------------------------------------
 * session_mysql.php
 * ------------------------------------------------------------------------
 * PHP4 MySQL Session Handler
 * Version 1.00
 * by Ying Zhang (ying@zippydesign.com)
 * Last Modified: May 21 2000
 *
 * ------------------------------------------------------------------------
 * TERMS OF USAGE:
 * ------------------------------------------------------------------------
 * You are free to use this library in any way you want, no warranties are
 * expressed or implied.  This works for me, but I don't guarantee that it
 * works for you, USE AT YOUR OWN RISK.
 *
 * While not required to do so, I would appreciate it if you would retain
 * this header information.  If you make any modifications or improvements,
 * please send them via email to Ying Zhang <ying@zippydesign.com>.
 *
 * ------------------------------------------------------------------------
 * DESCRIPTION:
 * ------------------------------------------------------------------------
 * This library tells the PHP4 session handler to write to a MySQL database
 * instead of creating individual files for each session.
 *
 * Create a new database in MySQL called "sessions" like so:
 *
 * CREATE TABLE sessions (
 *      sesskey char(32) not null,
 *      expiry int(11) unsigned not null,
 *      value text not null,
 *      PRIMARY KEY (sesskey)
 * );
 *
```

```
 * ------------------------------------------------------------------------
 * INSTALLATION:
 * ------------------------------------------------------------------------
 * Make sure you have MySQL support compiled into PHP4.  Then copy this
 * script to a directory that is accessible by the rest of your PHP
 * scripts.
 *
 * ------------------------------------------------------------------------
 * USAGE:
 * ------------------------------------------------------------------------
 * Include this file in your scripts before you call session_start(), you
 * don't have to do anything special after that.
 */

$SESS_DBHOST = "localhost";                  /* database server hostname */
$SESS_DBNAME = "sessions";             /* database name */
$SESS_DBUSER = "phpsession";           /* database user */
$SESS_DBPASS = "phpsession";           /* database password */

$SESS_DBH = "";
$SESS_LIFE = get_cfg_var("session.gc_maxlifetime");

function sess_open($save_path, $session_name) {
       global $SESS_DBHOST, $SESS_DBNAME, $SESS_DBUSER, $SESS_DBPASS, $SESS_DBH;

       if (! $SESS_DBH = mysql_pconnect($SESS_DBHOST, $SESS_DBUSER, $SESS_DBPASS))
{
              echo "<li>Can't connect to $SESS_DBHOST as $SESS_DBUSER";
              echo "<li>MySQL Error: ", mysql_error();
              die;
       }

       if (! mysql_select_db($SESS_DBNAME, $SESS_DBH)) {
              echo "<li>Unable to select database $SESS_DBNAME";
              die;
       }

       return true;
}

function sess_close() {
       return true;
}

function sess_read($key) {
       global $SESS_DBH, $SESS_LIFE;

       $qry = "SELECT value FROM sessions WHERE sesskey = '$key' AND expiry > " .
time();
       $qid = mysql_query($qry, $SESS_DBH);

       if (list($value) = mysql_fetch_row($qid)) {
              return $value;
       }

       return false;
}

function sess_write($key, $val) {
       global $SESS_DBH, $SESS_LIFE;

       $expiry = time() + $SESS_LIFE;
       $value = addslashes($val);
```

```
	$qry = "INSERT INTO sessions VALUES ('$key', $expiry, '$value')";
	$qid = mysql_query($qry, $SESS_DBH);

	if (! $qid) {
		$qry = "UPDATE sessions SET expiry = $expiry, value = '$value' WHERE
sesskey = '$key' AND expiry > " . time();
		$qid = mysql_query($qry, $SESS_DBH);
	}

	return $qid;
}

function sess_destroy($key) {
	global $SESS_DBH;

	$qry = "DELETE FROM sessions WHERE sesskey = '$key'";
	$qid = mysql_query($qry, $SESS_DBH);

	return $qid;
}

function sess_gc($maxlifetime) {
	global $SESS_DBH;

	$qry = "DELETE FROM sessions WHERE expiry < " . time();
	$qid = mysql_query($qry, $SESS_DBH);

	return mysql_affected_rows($SESS_DBH);
}

session_set_save_handler(
	"sess_open",
	"sess_close",
	"sess_read",
	"sess_write",
	"sess_destroy",
	"sess_gc");
?>
```

Y el script de testing es:

```
<?
/* ------------------------------------------------------------------------
 * test.php
 * ------------------------------------------------------------------------
 * PHP4 Customer Session Handler Test Script
 * Version 1.00
 * by Ying Zhang (ying@zippydesign.com)
 * Last Modified: May 21 2000
 */

/* default to DBM handler */
if (! isset($handler)) {
	$handler = "dbm";
}

/* default action is increment */
if (! isset($action)) {
	$action = "increment";
}

/* load up the appropriate session handling script, depending on the handler */
```

```
if ($handler == "dbm") {
	include("session_dbm.php");

} elseif ($handler == "mysql") {
	include("session_mysql.php");

} else {
	echo "<li>Unrecognized handler ($handler)";
	die;
}

/* start the session and register a simple counter */
session_start();
session_register("count");

/* figure out what we should do, depending on the action */
switch ($action) {
	case "increment" :
		$count = isset($count) ? $count + 1 : 0;
		break;

	case "destroy" :
		session_destroy();
		break;

	case "gc" :
		$maxlife = get_cfg_var("session.gc_maxlifetime");
		sess_gc($maxlife);
		break;

	default:
		echo "<li>Unknown action ($action)";
		break;
}
?>

<h1>Session Test Script</h1>
<ul>
<li>Handler: <b><?=$handler?></b>
<li>Action: <b><?=$action?></b>
<li>Count: <b><?=$count?></b>
</ul>

<hr size=1>
<form>
<table>
<tr>
	<td>Handler:</td>
	<td>
		<select name="handler">
		<option value="dbm">DBM</option>
		<option value="mysql">MySQL</option>
		</select>
	</td>
</tr>
<tr>
	<td>Action:</td>
	<td>
		<select name="action">
		<option value="increment">Increment</option>
		<option value="destroy">Session Destroy</option>
		<option value="gc">Force Garbage Collection</option>
		</select>
```

```
	</td>
</tr>
<tr>
	<td></td>
	<td><br><input type="submit"></td>
</tr>
</table>
</form>
```