Gestión de sesiones con PHP

I. SESIONES EN PHP

El soporte de sesiones en PHP es un mecanismo de programación, habilitado por defecto, que permite conservar cierta información sobre un usuario cuando éste pasa de una página web a otra dentro del sitio web.

Cuando se crea una sesión, con independencia de que se guarden o no datos de la misma, la información asociada se almacena en el servidor, en un archivo que se crea para la sesión y que se ubica en el directorio especificado por la opción `session.save_path` dentro del archivo `php.ini`.

Para almacenar datos asociados con una sesión se puede utilizar la variable global `$_SESSION`, que consiste en un array asociativo donde se guardan las variables de sesión disponibles para el script actual.

La creación de una variable de sesión se hace estableciendo un miembro apropiado en esta matriz. Por ejemplo:

```
$_SESSION['nombre_variable'];
```

Cuando se usa $_SESSION, se puede utilizar `unset()` para dejar de registrar una variable de sesión. Por ejemplo:

```
unset ($_SESSION['nombre_variable']);
```

Con la función `unset()` hay que tener cuidado y no ejecutarla sobre $_SESSION, ya que si se hace `unset($_SESSION)` se deshabilitará el registro de variables de sesión a través del array superglobal $_SESSION.

Algunas de las funciones existentes para el manejo de sesiones son:

- `session_start()`: crea una sesión o reanuda la actual basada en un identificador de sesión pasado mediante una petición GET o POST, o pasado mediante una cookie.
- `session_id()`: se utiliza para obtener o establecer el id de sesión para la sesión actual; si se especifica un argumento como identificador, reemplazará el identificador de sesión actual si se llama antes de session_start(); el identificador debe estar formado por caracteres en el rango a-z A-Z 0-9 , (coma) y - (menos), y debe contener al menos una letra; cuando se usan cookies de sesión, al especificar un identificador para session_id() se enviará siempre una nueva cookie cuando se llame a session_start(), sin importar si el identificador de sesión actual es idéntico al que se ha especificado. En PHP el identificador de una sesión se guarda en una cookie denominada PHPSESSID.
- `session_name()`: obtiene y/o establece el nombre de la sesión actual usado cookies y URLs; si se le da un argumento, que debería contener sólo caracteres alfanuméricos, actualizará con él el nombre de la sesión y devolverá el nombre antiguo de la sesión; el nombre de la sesión se reinicia al valor predeterminado almacenado en session.name en el momento de iniciar una petición, por tanto, se necesita llamar a session_name() para cada petición (y antes de llamar a session_start()).
- `session_destroy()`: destruye toda la información asociada con la sesión actual, pero no destruye ninguna de las variables globales asociadas con la sesión, ni destruye la cookie de sesión si la hubiera; para volver a utilizar las variables de sesión se debe llamar a session_start(); para destruir una sesión completamente, como por ejemplo al desconectar un usuario, debe destruirse el id de sesión; si se usa una cookie para propagar el id de sesión (comportamiento por defecto), entonces también debe borrarse la cookie de sesión (se puede utilizar setcookie() para eso.

- `session_status()`: se usa para devolver el estado de la sesión actual; los valores devueltos son: PHP_SESSION_DISABLED si las sesiones están deshabilitadas, PHP_SESSION_NONE si las sesiones están habilitadas, pero no existe ninguna y PHP_SESSION_ACTIVE si las sesiones están habilitadas, y existe una.
- `session_set_cookie_params()`: devuelve una matriz con la información de la cookie de sesión actual; la matriz contiene los siguientes elementos: lifetime (tiempo de vida de la cookie en segundos), path (ruta donde se almacena la información), domain (dominio de la cookie), secure (la cookie debería ser enviada solamente sobre conexiones seguras) y httponly (la cookie sólo puede ser accedida a través del protocolo HTTP).
- `session_save_path()`: devuelve la ruta del directorio actual usado para guardar la información de sesión; si se le da como argumento una ruta y la llamada se hace antes de session_start(), cambiará el directorio donde se guarda la información.
- `session_write_close()`: finaliza la sesión actual y almacena la información de la sesión; normalmente la información de la sesión se almacena después de que el script termine sin necesidad de llamar a session_write_close(), pero como la información de la sesión está bloqueada para prevenir escrituras concurrentes, el script sólo puede operar sobre una sesión cada vez (esto es particularmente importante cuando se usan sub-ventanas (frames) junto con sesiones, pues se observará que las sub-ventanas se cargan de una en una debido a ese bloqueo).

Otras funciones útiles para trabajar con sesiones son las que permiten manejar variables. Entre ellas se encuentra la función isset(), que permite comprobar si una variable ha sido inicializada.

Ejemplo:

En el caso siguiente se utilizan tres páginas web cuyos nombres de archivos se indican delante del código correspondiente. La primera página web sirve para que un usuario se identifique en una sesión y pueda tener acceso a partes reservadas en la segunda página y en la primera; además, enlaza con la tercera página donde se cierra la sesión del usuario. Para ejecutar este caso los archivos deben estar ubicados en el directorio htdocs, debe estar funcionando el servidor Apache y en el navegador debe escribirse:

http://localhost/sesionPagina_1.php

Archivo: sesionPagina_1.php

```
<?php
// Se abre una sesión
session_start();
?>
<!DOCTYPE html>
<html>
<head>
 <meta charset="UTF-8"/>
 <title>Sesiones: página 1</title>
</head>
<body>
 <?php
 if(isset($_SESSION['usuario'])){
        // Si estando el usuario identificado accede
        //a la página por segunda vez o más,
        // entonces se muestra un mensaje avisando de que la sesión está activa
        echo "<p>Ha vuelto a la página 1. Su sesión está activa y la inició como: " .
$_SESSION['usuario'] . "";
        echo "<p><a href='sesionPagina_3.php'>Cerrar sesión</a></p>";
 }else {
```

```
 ?>
       <! Si el usuario accede a la página sin estar identificado, se muestra un
formulario>
 <form action="sesionPagina_2.php" method="POST">
             <p>Nombre de usuario: <input type="text" placeholder="Escriba su
nombre" name="usuario" required/></p>
             <input type="submit" value="Pulse aquí para iniciar la sesión" />
 </form>
 <?php
 }
 ?>
 <p><a href="sesionPagina_2.php">Ir a la página 2</a></p>
</body>
</html>
```

Archivo: sesionPagina_2.php

```
<?php
 // Recupera la sesión abierta
 session_start();
?>
<!DOCTYPE html>
<html>
<head>
 <meta charset="UTF-8"/>
 <title>Sesiones: página 2</title>
</head>
<body>
 <?php
 if(isset($_POST['usuario'])){
       // Si el usuario accede a la página inmediatamente
       // después de identificarse se muestra un mensaje inicial
       $_SESSION['usuario'] = $_POST['usuario'];
       echo "Está en la página 2 y ha iniciado
             la sesión como: ".$_POST['usuario'];

 }else{
       if(isset($_SESSION['usuario'])){
 // Si estando el usuario identificado accede a la página por segunda vez o más,
 // entonces se muestra un mensaje diferente y la información de la sesión
       echo "Ha vuelto a la página 2. Su sesión sigue activa y la inició
como: ".$_SESSION['usuario'];
       echo "<p>Otra información de su sesión es la siguiente:";
             echo "<p>session_id: " . session_id();
             echo "<p>session_name: " . session_name();
             echo "<p>";
             print_r(session_get_cookie_params());

       }else{
             // Si el usuario accede a la página sin identificarse se
muestra un mensaje de aviso
             echo "Aviso: Acceso restringido. Debe iniciar sesión dando su
nombre.";
       }
 }
 ?>
 <p><a href="sesionPagina_1.php">Ir a la página 1</a></p>
</body>
</html>
```

Archivo: sesionPagina_3.php

```
<?php
 // Se recupera la sesión abierta y se cierra
 session_start();
 session_destroy();
?>
<!DOCTYPE html>
<html>
<head>
 <meta charset="UTF-8"/>
 <title>Sesiones: página 3</title>
</head>
<body>
 <p>Está en la página 3 y ha cerrado la sesión.</p>
 <p><a href="sesionPagina_1.php">Ir a la página 1</a></p>
</body>
</html>
```

III. COOKIES EN PHP

Las Cookies son un mecanismo por el cual se almacenan datos en el cliente remoto, de forma que se pueda rastrear o identificar a los usuarios que vuelven a conectarse. PHP soporta cookies HTTP de forma transparente.

Una cookie es un fragmento de información que un navegador web almacena en el disco duro del visitante de una página web. La información se almacena a petición del servidor web, y puede ser recuperada por el servidor web en visitas que se hagan posteriormente a la misma página web.

Las cookies resuelven un problema del protocolo HTTP, que no es capaz de mantener información persistente entre diferentes peticiones. Además, permiten compartir información entre distintas páginas de un sitio web o incluso en la misma página web pero en diferentes instantes de tiempo.

La cookies se pueden configurar usando la función `setcookie`(), que permite asignar valor a una cookie, o `setrawcookie`(), que funciona como `setcookie`() excepto que el valor de la cookie no se codificará automáticamente cuando se envíe al navegador.

El formato de `setcookie`() es:

```
bool setcookie ( string $name [, string $value [, int $expire = 0 [, string $path [, string $domain [, bool $secure = false [, bool $httponly = false ]]]]]] )
```

En el caso anterior, exceptuando el argumento name, los demás argumentos son opcionales. Si se quiere saltar un argumento, se puede utilizar un string vacío (""), aunque como el argumento expire es un entero, para pasarlo por alto debe utilizarse un cero (0).

El significado de los argumentos es el siguiente:

- **`name`**: nombre de la cookie.
- **`value`**: valor de la cookie; este valor se guarda en el equipo del cliente, por lo que no debería almacenar información sensible.
- **`expire`**: tiempo en el que caduca la cookie; se expresa en número de segundos, siendo habitual utilizar la función `time`() más el número de segundos que se quiere que dure la cookie; también se podría utilizar la función `mktime`(); si se pone 0, o se omite, la cookie caducará al final de la sesión (al cerrarse el navegador).
- **`path`**: ruta dentro del servidor en la que la cookie estará disponible; si se utiliza '/', la cookie estará disponible en todo el domain; el valor por defecto es el directorio actual en donde se está configurando la cookie.
- **`domain`**: dominio para el cual la cookie está disponible; las cookies disponibles en un dominio inferior estarán disponibles en dominios superiores.
- **`secure`**: indica que la cookie sólo debería transmitirse por una conexión segura HTTPS desde el cliente; cuando se configura como TRUE, la cookie sólo se creará si es que existe una conexión segura; del lado del servidor, depende del programador el enviar este tipo de cookies solamente a través de conexiones seguras (por ejemplo, con $_SERVER["HTTPS"]).
- **`httponly`**: puede tomar los valores TRUE o FALSE; cuando es TRUE la cookie será accesible sólo a través del protocolo HTTP, por lo que la cookie no será accesible por lenguajes de scripting, como JavaScript.

Para borrar una cookie también se utiliza `setcookie`(), pero asignando a la cookie una fecha de caducidad (expire) ya pasada, es decir, una fecha anterior a la actual.

Las Cookies son parte de la cabecera de HTTP, por lo que la llamada a `setcookie`() se hará antes de que se envíe al navegador cualquier otra salida, es decir, las cookies se deben enviar antes que `<html>`, `<head>` o un simple espacio en blanco. Esta limitación es la misma que tiene la función `header`() que se utiliza para enviar encabezados HTTP sin formato.

Sin embargo, se pueden usar funciones del buffer de salida para retrasar la salida del script hasta que se halla decidido enviar o no cookies o enviar cualquier otro header. Así se podría enviar un encabezado en cualquier momento, pero con el coste adicional de almacenar en el servidor toda la salida antes de enviarla al cliente y no recibirla en el cliente hasta que el código PHP no haya terminado por completo. Entre las funciones de manejo del buffer que se pueden utilizar en una página web se encuentran ob_start(), que activa el almacenamiento del buffer, y ob_end_flush(), que produce la salida de lo almacenado; también puede ser necesario configurar la directiva output_buffering en el archivo php.ini.

Algunos envíos de cookies desde el cliente serán incluidos automáticamente en la matriz global $_COOKIE si la directiva variables_order en el archivo php.ini contiene C (de Cookie). Para asignar múltiples valores a una cookie, únicamente es necesario agregar [] al nombre de la cookie. Para recuperar el valor de una cookie se emplea el array predefinido $_COOKIE con el nombre de la cookie como índice. También se puede emplear $_REQUEST, que contiene la unión de $_COOKIE, $_POST y $_GET.

Como se ha dicho en el apartado anterior, los datos asociados a una sesión se almacenan en el servidor y nunca en el cliente. En la mayoría de las tecnologías de web, las sesiones se implementan mediante una cookie que almacena un valor que permite identificar al usuario en el servidor web cada vez que pasa de una página web a otra. En el servidor web se almacenan todos los datos de la sesión y se accede a ellos cada vez que se pasa de página gracias al identificador almacenado en la cookie.

En Mozilla Firefox podemos ver las cookies que hay almacenadas, con su valor y fecha de caducidad accediendo a:

`Opciones > Opciones > Privacidad > Mostrar cookies.`

Ejemplo:

El siguiente código permite crear una cookie que caduca en una hora.

Archivo: cookieCrear.php

```
<?php
 $value = 'Cookie de Sistemas de Información Basados en Web';
 setcookie("pruebaCookie", $value, time()+3600);  /* expira en una hora */
 echo "Se ha creado una cookie denominada pruebaCookie.";
?>
```

Ejemplo:

Mostrar el valor de la cookie creada anteriormente sabiendo su nombre.

Archivo: cookieMostrar.php

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8"/>
  <title>Mostrar una cookie conociendo su nombre</title>
</head>
<body>
  <?php
  // Se muestra el valor de la cookie creada anteriormente
  echo "<p>El valor de la cookie pruebaCookie es: " . $_COOKIE["pruebaCookie"] . "</p>";
  ?>
</body>
</html>
```

Ejemplo:

Borrar la cookie creada anteriormente sabiendo su nombre.

Archivo: cookieBorrar.php

```
<?php
 // Para eliminar la cookie se establece su caducidad con tiempo negativo
 setcookie ("pruebaCookie", "", -1);
 echo "Se ha eliminado una cookie denominada pruebaCookie.";
?>
```

Ejemplo:

Mostrar las cookies existentes, aunque no se conozcan sus nombres.

Archivo: cookieMostrarTodas.php

```
<!DOCTYPE html>
<html>
<head>
	<meta charset="UTF-8"/>
	<title>Mostrar todas las cookies</title>
</head>
<body>
	<?php

		// Se muestra de dos formas diferentes la matriz con todas las cookies existentes
		echo "<h1>Se muestra de dos formas la matriz de las cookies existentes:</h1>\n";
		echo "Formato 1:\n";
		print_r ($_COOKIE);
		echo "<p>Formato 2: ";
		var_dump ($_COOKIE);
		echo "</p>";

		// Ahora se muestra la lista de nombres y valores de las cookies existentes
		echo "<h1>La lista de nombres y valores de las cookies existentes es:</h1>\n";
		while (list($clave, $valor) = each ($_COOKIE)) {
			echo "<p>Cookie: " . $clave . " => " . "Valor: " . $valor . "</p>";
		}

		// Si únicamente hacen falta los valores de las cookies existentes se pueden obtener de otro modo
		echo "<h1>La lista de valores de las cookies existentes es:</h1>\n";
		foreach ($_COOKIE as $valor) {
			echo "<p>" . $valor . "</p>";
		}
	?>
</body>
</html>
```

Ejemplo:

En el caso siguiente se utilizan tres archivos con el fin de crear una cookie a partir de un formulario de HTML y mostrar el valor correspondiente. Para ejecutar este caso en el navegador debe escribirse lo siguiente:

Para crear la cookie:

http://localhost/cookieFormulario.html

Para ver su valor:

http://localhost/verCookie.php

Archivo: cookieFormulario.html

```
<html>
<head>
 <title>Uso de cookie con un formulario</title>
</head>
<body>
 <H1>Ejemplo de uso de cookie con un formulario</H1>
 <form action="activarCookie.php" method="GET">
       <p>Nombre de usuario: <input type="text" placeholder="Escriba su
nombre" name="usuario" required/></p>
       <input type="submit" value="Pulse aqu&iacute; para enviar" />
 </form>
</body>
</html>
```

Archivo: activarCookie.php

```
<?php
  setcookie("pruebaFormulario", $_GET['usuario'], time()+ 600, "/", "");
?>

<html>
<head>
  <title>Activar cookie</title>
</head>
<body>
  <H1>Se establece la cookie</H1>
  <p>Se ha creado una cookie denominada <b>pruebaFormulario</b> cuyo valor es <b>
  <?php print $_GET['usuario']; ?> </b> y que caduca en diez minutos.</p>
</body>
</html>
```

Archivo: verCookie.php

```
<html>
<head>
 <title>Ver cookie</title>
</head>
<body>
 <H1>Se muestra el valor de la cookie</H1>
 <p>El valor de la cookie denominada <b>cookieFormulario</b> es: <b>
 <?php print $_COOKIE['pruebaFormulario']; ?> </b></p>
</body>
</html>
```

IV. ENLACES INTERESANTES

http://www.php.net/manual/es/features.sessions.php

http://www.php.net/manual/es/features.cookies.php

http://www.php.net/manual/es/book.var.php