Tema 7

Sesiones y Cookies

Sabemos que el protocolo HTTP, encargado de determinar la manera en que cliente y servidor se comunican, es un protocolo sin estado. Esto significa que dicho protocolo es incapaz de determinar el estado entre dos transacciones diferentes.

Supongamos que dos scripts PHP diferentes desean intercambiar información entre ellos sin necesidad de solicitar información al usuario. Teniendo en cuenta que HTTP es un protocolo sin estado, no tendríamos forma de enviar la información desde el primero al segundo, a no ser que utilicemos formularios. Hacer eso sería bastante engorroso. Necesitaríamos una herramienta que nos permita realizar esta tarea fácilmente y es en este punto en el que PHP introduce los conceptos de **cookie** y **sesión**.

Cabeceras HTTP

Tanto **cookies** como **sesiones** hacen uso de las **cabeceras HTTP**. Si recordamos, en temas anteriores se explicó que los mensajes transmitidos entre cliente y servidor están constituidos por dos partes: **cuerpo** y **cabecera**.

PHP permite modificar la información por defecto que es enviada en la cabecera de un mensaje. Esta información es recibida y procesada por el cliente antes de mostrar el contenido en el navegador. La función encargada de esta tarea se llama **header** y permite informar al navegador del código de estado HTTP, codificación utilizada, tipo de archivo, etc.

Esta función puede recibir hasta tres parámetros. El primero es obligatorio y define el encabezado que se va a incorporar al mensaje. Aunque podemos utilizarlo para diferentes situaciones, a lo largo de este curso emplearemos esta función para dos en concreto. Comenzamos por las redirecciones.

```php
header("Location: foo.php") ;
```

Básicamente, esto sería suficiente. No obstante, opcionalmente podemos utilizar el tercer parámetro para ofrecer más información sobre la operación de redirección a través de un código HTTP.

```php
header("Location: foo.php?opcion=borrar&id=25", true, 301) ;
```

El código **301** (**Moved Permanently**) indica que la comunicación con el servidor se ha producido correctamente, pero que el recurso solicitado ha sido movido a otra dirección de manera permanente. El segundo parámetro indicará que, en caso de existir previamente, el encabezado será reemplazado por el que le hemos proporcionado a la función; utilizar **false** como segundo parámetro forzará la convivencia de encabezados del mismo tipo.

Debemos recordar que **no debemos utilizar esta función una vez se ha volcado contenido en pantalla**. Esto se debe a que, como sabemos, el servidor enviará la cabecera antes que el contenido generado por el script. De esta manera, la función **header** no podrá modificar la cabecera del mensaje si ya hemos escrito en el documento código HTML, líneas en blanco o cualquier mensaje desde PHP, puesto que la cabecera ya habrá sido enviada.

PHP nos permite generar archivos de todo tipo, como imágenes, documentos PDF, etc. Para que el navegador reconozca el tipo de archivo debemos enviar el tipo **MIME** (**Multipurpose Internet Mail Extensions**), que permitirá al navegador del cliente interpretar de forma apropiada el contenido recibido. Utilizaremos también la función **header** para este cometido.

```
header("Content-Type: application/xml") ;
```

La línea anterior indicará al navegador que la información que estamos generando desde nuestro script deberá interpretarse como código XML. Añade el siguiente fragmento de código tras la línea anterior y comprueba la diferencia entre enviar o no la cabecera.

```
<alumno>
  <matricula>M007</matricula>
  <nombre>Jorge</nombre>
  <apellidos>Jiménez Jaén</apellidos>
  <edad>18</edad>
  <datos_academicos>
    <asignatura>Desarrollo Web en Entorno Servidor</asignatura>
    <estado>aprobada</estado>
    <nota>9.5</nota>
  </datos_academicos>
</alumno>
```

Si enviamos la cabecera estaremos indicando al navegador que debe interpretar el contenido como un documento XML. De esta manera, el contenido se mostrará formateado de manera apropiada en pantalla. Sin embargo, si no enviamos la cabecera, la información enviada al cliente se mostrará como texto plano en el navegador, ya que éste no sabrá cómo interpretarlo.

PHP nos permite obtener información sobre la información que maneja el protocolo HTTP a través de las variables **$_SERVER** y **$_REQUEST**. Ambos son arrays asociativos y, mientras

el primero de ellos contiene información tal como las cabeceras, rutas y ubicación del script, **$_REQUEST** agrupa los datos de las variables superglobales **$_GET**, **$_POST** y **$_COOKIE**.

Cookies (galletas)

Si eres un internauta asiduo estoy seguro que ya conocerás el concepto de **cookie**. Básicamente, podemos definir una **cookie** como una pequeña y dulce cantidad de información que puede ser almacenada en la máquina cliente. Esto las hace poco apropiadas para almacenar información sensible; además es posible que el usuario haya bloqueado el almacenamiento de cookies.

```
setcookie("nombre", "valor", time() + 3600, "/foo", "midominio.com") ;
```

La llamada anterior a **setcookie** creará en la máquina cliente una cookie que se llamará como se indica y guardará el **valor** dado. Se ha definido para que **expire** en una hora (la función **time** devuelve la hora UNIX actual y le sumamos 3600 segundos) y será visible únicamente en el **dominio** indicado y para los scripts del servidor que encontremos en la ruta **/foo**. Todo esto puede hacerse también enviado la siguiente información mediante la función **header**.

```
Set-Cookie: NOMBRE = VALOR; [expires = FECHA ;] [path = RUTA ;] [domain = DOMINIO ;]
```

Las cookies deben ser gestionadas antes de que se envíe cualquier información al navegador, ya que el tratamiento de las mismas se realiza en la cabecera del mensaje HTTP. Debido a esto, los datos de un cookie tienen prioridad sobre los enviados desde un formulario. De esta manera, cuando ambos hacen uso de un mismo identificador, prevalecerá siempre el de la *galleta*.

Cuando volvamos a visitar nuestra página tendremos acceso a las cookies creadas a través de la variable **$_COOKIE** que, al igual que otras superglobales es un array asociativo que contiene información sobre cada una de las galletas almacenadas en la máquina cliente. Debemos recordar que para borrar una cookie previamente creada, no tendremos más remedio que utilizar nuevamente la función **setcookie** especificando el mismo nombre y haciéndola expirar en algún momento del pasado.

Sesiones

Como vimos anteriormente, el uso de cookies plantea una serie de inconvenientes que se resuelven fácilmente mediante las **variables de sesión.** Éstas se almacenarán en el propio

servidor y estarán disponibles durante todo el tiempo que sea necesario. El ciclo de vida de estas variables finaliza en el mismo instante en que el usuario decide **terminar su sesión**, se **cierra el navegador** o se destruyen automáticamente cuando ha transcurrido un período de tiempo sin que haya interacción entre el usuario y la aplicación.

Cada una de estas variables se encontrará vinculada a una única **sesión de usuario**, que queda definida unívocamente por un identificador que deberemos preservar a lo largo de la sesión. El ID de sesión es enviado al navegador a través de una **cookie de sesión**, aunque en ocasiones algunas aplicaciones optan por enviarlo codificado en la misma URL, lo cual no es muy aconsejable ya que aumentan los riesgos de seguridad de nuestra aplicación.

Creación y flujo de una sesión

Cuando una sesión se inicia, PHP puede recuperar una sesión existente, o bien crear una nueva. La información asociada a una sesión se almacenará en el array superglobal **$_SESSION**. Accederemos a dicho array para poder tener acceso a las diferentes variables de sesión que hayamos decidido registrar para, finalmente eliminarlas y destruir la sesión. Este flujo de ejecución no tiene por qué encontrarse necesariamente en un único script PHP.

Cuando deseemos iniciar una sesión tendremos que hacer uso explícito de la función **session_start()**. La llamada a esta función debe ser lo primero que hagamos en nuestro script PHP, en otro caso, no se almacenarán ninguna de las variables de sesión que defina nuestra aplicación.

```
session_start( ... ) ;
```

La llamada a esta función creará automáticamente una nueva sesión y generará un nuevo identificador, o bien retoma una sesión existente gracias al ID propagado. De manera optativa podemos proporcionarle a la función un array de opciones que sobrescribirá los valores por defecto establecidos para las directivas de configuración para sesiones que encontramos en el archivo **php.ini**.

```
session_start(["cookie_lifetime" => 86400]) ;
```

En el ejemplo anterior, se modificará además el valor por defecto para la directiva **cookie_lifetime** que permite especificar el tiempo de vida (en segundos) de las cookies de sesión.

Creación y acceso a la variable de sesión

Crear y acceder a una variable de sesión es tan simple como utilizar un array asociativo. Tal y como se dijo anteriormente, el encargado de almacenar la información asociada a una sesión es el array **$_SESSION**, este no será más que un almacén de variables de sesión y

sus correspondientes valores, que crearemos de igual manera a como hiciéramos en temas anteriores con este tipo de arrays.

```
$_SESSION["usuario"] = "admin" ;
```

La línea anterior registra la variable de sesión usuario y le asocia la cadena **admin**. Acceder a dicho valor, una vez iniciada la sesión, será tan sencillo como acceder al correspondiente índice del array.

```
$usuario = $_SESSION["usuario"] ;
```

Las variables de sesión registradas permanecerán en memoria hasta que la sesión termine, o hasta que se destruyan, para lo cual haremos uso de la función **unset**. Si, por ejemplo, deseamos destruir la variable usuario creada anteriormente bastará con hacer lo siguiente.

```
unset($_SESSION["usuario"]) ;
```

Debes recordar que, destruir por completo la variable **$_SESSION** deshabilita el registro de variables de sesión.

Destrucción de una sesión

Destruir una sesión es tan sencillo como utilizar la función **session_destroy**. Sin embargo, una sesión no habrá sido destruida completamente hasta que también hayan sido destruidos su ID y la cookie de sesión que hayamos empleado para su propagación. En ese instante, podemos considerar al usuario como *desconectado*. Si lo deseamos, podemos destruir de una vez todas las variables de sesión.

```
$_SESSION = [] ;
```

El uso de la función **session_unset** se desaconseja completamente, excepto para códigos que se ejecuten sobre servidores con versiones antiguas de PHP.

Otras funciones

Aunque lo visto nos será más que suficiente para desarrollar nuestras aplicaciones, PHP incorpora bastantes funciones para trabajar con sesiones y cookies. Algunas de estas se detallan en la siguiente tabla, aunque deberemos acceder a la documentación oficial del lenguaje para obtener información más amplia sobre su funcionamiento.

Función	Descripción
session_abort()	Desecha los cambios realizados en el array

	$_SESSION y finaliza la sesión.
`session_encode()`	Codifica en una cadena los datos de la sesión actual contenidos en **$_SESSION**.
`session_decode( $data )`	Realiza la operación inversa a la función anterior. Decodifica un conjunto de datos de sesión almacenados en forma de cadena, y rellena automáticamente con ellos la variable **$_SESSION**.
`session_gc()`	Se utiliza para realizar una recolección de basura de los datos de sesión.
`session_create_id( [$pref] )`	La función se introduce en la versión 7 de PHP y se utiliza para crear nuevos identificadores de sesión libres de colisiones para la sesión actual. Opcionalmente podemos especificar un prefijo para los nombres de sesión. Están permitidos únicamente caracteres alfanuméricos, la coma y el guión.
`session_reset()`	Reinicia el array de sesión con los valores originales. Requiere que se encuentre activa una sesión.
`session_set_cookie_params(...)`	Nos permite establecer los parámetros de las cookies de sesión. Visítese el manual de referencia oficial para más información.
`session_get_cookie_params(...)`	Obtiene el valor de los parámetros de las cookies de sesión. Visítese el manual de referencia oficial para más información.

Bibliografía

PHP.net
Guía oficial del lenguaje

PHP and MySQL Web Development
Luke Welling, Laura Thompson
Developer's Library

Learning PHP7
Antonio López
Packt Publishing