

INTRODUCCIÓN A LA PROGRAMACIÓN CON PHP

SESIONES

PRÁCTICA Nº 7

¿Qué son las sesiones?

Se pueden definir como:

Un conjunto de variables almacenadas en el servidor; única por cada entidad que accede a la página.

o también como:

Estructura de datos almacenadas en el servidor, que ofrecen información del seguimiento del cliente, ésta información es individual para cada cliente.

Las sesiones nos permiten registrar un número arbitrario de variables que se conservan durante toda la visita de un usuario a una página web.

Dichas variables pueden ser diferentes para cada usuario, ya que están referenciadas por un identificador único que se le asigna a cada visitante.

En otras palabras, una sesión es una manera de almacenar variables de manera temporal, semejante a una cookie, pero con ciertas diferencias: las cookies se almacenan en la PC del usuario y pueden desactivarse; las sesiones, en cambio, se almacenan temporalmente en el servidor, en un fichero que se crea en el momento en que almacenamos la variable.

Generalmente las sesiones tienen 5 métodos principales:

- ❖ Abrir sesión
- ❖ Definir variable de sesión
- ❖ Definir el valor de una variable en sesión
- ❖ Obtener el valor de una variable en sesión
- ❖ Cerrar sesión

¿Dónde se usan?

Las sesiones son estructuras que se usan en aplicaciones cliente-servidor, tales como en ftp, telnet... aplicaciones web.

Protocolo de red que permite acceder a otra máquina para manejarla remotamente como si estuviéramos sentados delante de ella

File Transfer Protocol usado en internet.
Permite transferir archivos locales hacia un servidor web

¿Para qué sirven las sesiones?

Las sesiones se utilizan como método para conservar ciertos datos a lo largo de los subsiguientes accesos.

En las páginas webs permite construir aplicaciones más personalizadas e incrementar el atractivo de una página web.

¿Cómo sabe php cual es la sesión activa?

PHP utiliza dos métodos para identificar la sesión:

- *mediante una cookie con un identificador único de la sesión,*
- *o bien mediante un parámetro: SID, el cual contiene éste identificador.*

Alternativamente, se puede usar la constante SID, que está definida si la sesión se inició.

Si el cliente no envía una cookie de sesión apropiada, tiene la forma session_name=session_id.

De otro modo se desarrolla en una cadena vacía. Así, se puede embeberla incondicionalmente dentro de las URL.

Constantes predefinidas:

Estas constantes están definidas por esta extensión y estarán disponibles sólo cuando la extensión haya sido compilada con PHP, o bien sea cargada dinámicamente en ejecución.

- *SID ([string](#)):*

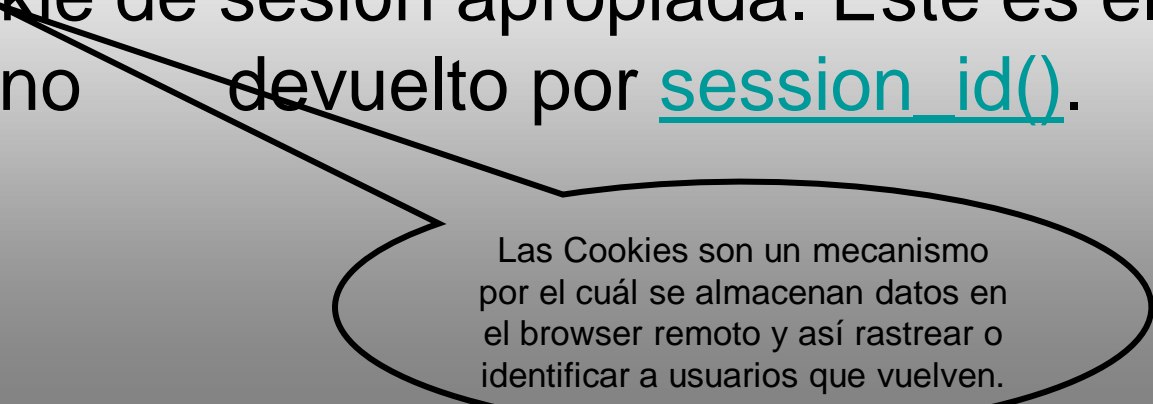
Constante que contiene el nombre de la sesión y el ID de sesión en la forma "name=ID" o una cadena vacía si el ID de sesión fue establecido en una cookie de sesión apropiada.

Constantes predefinidas

Son constantes que están definidas en la extensión y estarán disponibles sólo cuando la extensión haya sido compilada con PHP, o bien sea cargada dinámicamente en ejecución.

SID

Constante que contiene el nombre de la sesión y el ID de sesión en la forma **"*name=ID*"** o una cadena vacía si el ID de sesión fue establecido en una cookie de sesión apropiada. Este es el mismo id que uno devuelto por [session_id\(\)](#).



Las Cookies son un mecanismo por el cuál se almacenan datos en el browser remoto y así rastrear o identificar a usuarios que vuelven.

¿Por qué son seguras?

A diferencia de las cookies, las cuales guardan la información en el cliente, los valores, se guardan en el servidor, de forma inaccesible al cliente.

Por ejemplo; puedes tener una variable en la sesión actual que sea 'logueado', con el valor 'true', ésta información será accesible a todas las páginas PHP, y cada sesión tendrá sus propios valores, (cada explorador abierto en cada ordenador tiene su sesión propia (siempre y cuando se abra), con ésto tienes la certeza de estar logueado, ya que si fueran cookies el cliente podría manipular el valor y acceder a ellas.

Otra forma de saber si se está logueado es colocar en las cookies valores con el nombre del usuario y el password, y verificar cada vez estos valores,... a parte de que es mucho mas rápido obtenerlos de la sesión, esta información pasaría constantemente de una forma poco segura y además se quedaría guardado en el cliente, y otras personas podrían verlo.

Ejemplo del Carrito de Compras

`session_start()`

Creamos la sesión si no existe, o la retomamos si ya ha sido creada

Sintaxis:

`bool session_start (void)`



extract(array asociativo)

Crea de **forma automática** variables con los nombres de todos los índices del array y les asigna el valor de ese elemento del array.

Es decir, si hemos recibido valores como `$_GET['nombre1']=7` y `$_GET['nombre2']='pepe'` e incluimos en el script la función `extract($_GET)`

PHP creará de forma automática las variables `$nombre1=7` y `$nombre2='pepe'`.

Mediante `extract($_REQUEST)` obtendrían los mismos resultados. En definitiva, la función **extract()** no hace otra cosa que **crear variables PHP con idéntico nombre al indicado en la petición.**

Hay configuraciones de PHP en las que no es necesario utilizar la función **extract()**.

Cuando el fichero **php.ini** tiene configurada la directiva `register_globals=ON` las variables del tipo: **\$nombre1=valor1**, **\$nombre2=valor2**, etcétera se crearían de forma automática al ser recibida la petición del cliente.

extract(\$_REQUEST);

extract — Importar variables a la tabla de símbolos actual desde un array

La función extract toma las claves de una matriz asociativa y las convierte en nombres de variable, asignándoles a esas variables valores iguales a los que tenía asociados en la matriz.

Es decir, convierte en nuestro ejemplo a `$_GET['id']` en `$id`, sin que tengamos que tomarnos el trabajo de escribir `$id=$_GET['ID'];`

Un array asociativo que por defecto contiene el contenido de `$_GET`, `$_POST` y `$_COOKIE`.

Encriptación MD5 con PHP

La información que se guarda en la base de datos a través de los sitios web tiene que tener algún tipo de protección.

Es por ello que algunos campos se guardan encriptados en la base de datos, principalmente cuando una página requiere el nombre de usuario y contraseña, esta última se encripta y se guarda en la Base de datos.

En PHP se utiliza la función MD5 (Message Digest 5), que es una función hash irreversible (de un sólo sentido) , es decir, encripta el password tecleado por el usuario y es imposible que partiendo desde la cadena encriptada se vuelva a la contraseña origen. Por esto mismo no hay problema de que alguien pueda acceder al campo encriptado de la base de datos.

Como en la base de datos se guarda la contraseña encriptada, cuando un usuario quiere acceder, habrá que realizar una comparación entre el password que introduce encriptado en MD5, y lo que tenemos en la base de datos, (que es la contraseña encriptada en MD5), si coincide se le permite el acceso, si no, se rechaza.



MD5 se utiliza también para que cuando un usuario olvida su password, si quiere recuperar la contraseña se le pide que introduzca por ejemplo el correo, y se le envía un mail con una URL tal que si entra en ella genere una nueva contraseña que se le indica al usuario y se reescribe en md5 en la base de datos (borrando la anterior contraseña).

Hay que tener en cuenta que esto no es 100% seguro, puesto que la contraseña se encripta en el servidor, entonces al enviar la contraseña desde el cliente al servidor podría ser interceptada.

Para hacernos una idea, el algoritmo MD5 convierte el mensaje en un bloque múltiplo de 512 bits, (si hace falta añadirá bits por el final). Luego toma el primer bloque de 512 bits del mensaje y realiza diversas operaciones lógicas con los 128 bits de cuatro vectores iniciales ABCD de 32 bits cada uno. (Dichos vectores tendrán el valor inicial que nosotros queramos).

Como resultado obtiene una salida de 128 bits que se convierte en el nuevo conjunto de los 4 vectores ABCD. Se repite el algoritmo hasta procesar el último bloque del mensaje. Al terminar, el algoritmo devuelve los últimos 128 bits de estas operaciones.

La definición de la función md5 en PHP es

```
string md5(string cad)
```

ob_start

Activa el almacenamiento en búfer de salida. Mientras dicho almacenamiento esté activo, no se enviará ninguna salida desde el script (aparte de cabeceras), en su lugar la salida se almacenará en un búfer interno.

b_end_flush

Vuelca (enviar) el búfer de salida y deshabilitar el almacenamiento en el mismo

Las funciones `ob_start` y `ob_end_flush` te permiten elegir en qué momento enviar el resultado de un script al navegador.

Si no las utilizamos estamos obligados a que nuestra primera línea de código sea `session_start()` u obtendremos un error

