COOKIES Y SESIONES

Cookies

Una *cookie* es una especie de variable que permite almacenar localmente informaciones diversas. Se almacenan en un fichero que puede contener diferentes *cookies*. Su finalidad es la de poder guardar información de un visitante, dejándola en su ordenador, para poderlas recuperar y utilizar en futuras visitas (si ya se ha visitado la página o no, fecha y hora de la última visita, etc.). Todas las *cookies* de una misma web se guardan secuencialmente en un mismo fichero de texto.

Dependiendo del Sistema Operativo en el que operemos y de su versión, y asimismo también teniendo en cuenta el navegador, encontraremos las *cookies* en un lugar u otro.

La creación de *cookies* en PHP se hace a través de la función **setcookie()** .
En suma, para crear una *cookie* nos basta con darle un *nombre* , un *contenido* y un *momento de expiración* de la misma.

Tal y como se indica en la descripción de esta función, hay que tener presente que las cookies deben enviarse *antes* de mandar cualquier otra cabecera (esta es una restricción de las cookies, no de PHP). **Esto requiere que sitúe las llamadas a esta función antes de cualquier etiqueta <*html*> o <*head*>.**

El siguiente ejemplo pretende ilustrar su definición.

```
<?php
// Momento de expiración de la cookie
// que será dentro de un año a partir de la hora
// actual que devuelve la función 'time()'
$expira = 365*24*3600;
// Creamos la cookie
setcookie("prueba","este es el contenido",time()+$expira);
echo "<h2>\"Cookie\" creada </h2><br>";
echo "Comprueba que la cookie ha sido creada";
?>
```

Como complemento al script propuesto, implementamos otro que acceda a la *cookie*.

```
<?php
// Recuperamos el nombre de usuario de 2 formas
// 1) mediante el array '_COOKIE'
echo "Hola ".$_COOKIE['prueba']." <br><br>";
// 2) utilizando la variable del mismo nombre directamente
//echo "Hola $prueba <br><br>";
?>
```

Explicación de setCookie
setCookie(nombre, [valor [, fecha expiración, [, ruta [, dominio [. Seguridad]]]]])

Nombre: Es el nombre que se le asigna a la cookie. Es el único valor obligatorio.
Valor: Son los datos que almacenamos en el navegador visitante.
Fecha de expiración: En segundos.
Ruta: Este parámetro nos servirá en el caso de que queramos restringir el acceso de la cookie a una determinada ruta de nuestro servidor .
Dominio: Es como el anterior (ruta), pero en vez de restringir o dar permisos a carpetas lo hace con dominios-
Seguridad: Este parámetro es el que determina la seguridad de la cookie, tiene 2 valores, 0 y 1. Con el valor 1 indica que la cookie sólo se enviará vía servidor seguro (SSL) y con el 0, indicará que podrá ser mandada con cualquier conexión.

La función setcookie toma esencialmente 3 parámetros. El primero es el nombre de la cookie, el segundo el valor y el tercero el tiempo de expiración, es decir, en que fecha deberá el navegador del usuario borrar esa cookie.

Por ejemplo:

```
setcookie("visitas",1,time()+30*24*60*60);
```

Con esta sentencia pondremos una cookie llamada visitas, con el valor 1 y que expirará dentro de 30 días. También tenemos que tener en cuenta que la cookie no la tendremos disponible hasta que el usuario recargue la página.
Para leer una cookie lo podemos hacer con la variable $_COOKIE['nombrecookie'].

```
Por ejemplo:
```

```
<?php
if(isset($_COOKIE['visitas'])) {
$_COOKIE['visitas']++;
setcookie("visitas",$_COOKIE['visitas'],time()+30*24*60*60);
} else {
setcookie("visitas",1,time()+30*24*60*60);
}
echo "Contador: ".$_COOKIE['visitas']." <br><br>";
?>
```

Con este pequeño script lo que hemos hecho es un contador de visitas por usuario. De esta forma sabremos cuantas visitas ha hecho el usuario a la página donde pongamos este código.

Implementa el contador de visitas.

Sesiones

Existen casos de aplicaciones Web en las que, dadas sus características, se hace necesario disponer de la posibilidad de mantener cierta información a lo largo del proceso de navegación entre las páginas que la integran.

Ejemplos de esto podrían ser tanto la banca on-line, en la que cada usuario dispone de un área de navegación personalizada, como el típico "carrito de la compra", en el que el usuario selecciona y acumula una serie de artículos en el proceso de navegación entre una serie de páginas-catálogo. En estos casos es necesario relacionar las páginas que forman parte de esa área de navegación dependiente de una información común.

Visto que las comunicaciones que se establecen a través del protocolo HTTP son independientes las unas de las otras, lo cual cierra la posibilidad de establecer cualquier tipo de comunicación entre una serie de páginas, las sesiones vienen a cubrir este tipo de requerimiento, ofreciendo un mecanismo para el mantenimiento ubicuo de esa información compartida, al margen del funcionamiento de este tipo de protocolo.

¿Cómo implementa PHP las sesiones?

Una sesión, a groso modo, no es más que un conjunto de variables (que pueden ser concebidas como variables "globales") que PHP gestiona de manera transparente al usuario.

Cada sesión que se inicia debe diferenciarse y ser independiente del resto (ya que una página puede ser accedida al mismo tiempo por diversos usuarios), por lo que PHP asigna un identificador por cada sesión que se inicie, siendo este identificador una secuencia alfanumérica generada automáticamente. PHP dispone de un método mixto de almacenamiento de la información relativa a una sesión:

Por un lado, el identificador de sesión se almacena mediante "cookies".
Problema: el usuario podría tenerlas desactivadas, lo cual nos obligaría a tener que pasar la información "manualmente", o mediante el *query_string* (en etiquetas <A>) o mediante un campo de tipo "hidden" (si venimos de un formulario).

Por otro, el resto de información se almacena en un fichero en el servidor (por defecto, en el directorio "./tmp" que deberá crearse si no existe).

Por tanto:
Se aprovechan las cookies para el reenvío del identificador en cada página.
En caso de no aceptar cookies, se garantiza el funcionamiento recurriendo al paso explícito de la información.

¿Cómo se gestionan las sesiones desde código?
La gestión de sesiones se realiza de manera sencilla a través de un juego de funciones

Existen **3** maneras de iniciar/prolongar una sesión:

- Invocando a la función **session_start()**

```
<?php
session_start();
echo "He inicializado la sesión";
?>
```

- A través de de la función **session_register()** (creación de variables).

```
<?php
// Creamos la variable de sesión 'contador'
session_register('contador');
// La página enlaza consigo misma y muestra
// el identificador de sesión (variable interna $SID)
// y el valor del contador actualizado
echo '<a href="'.$PHP_SELF.'?'.$SID.'">Contador vale:
'.++$contador.'</a>';
?>
```

- Activando la directiva **session.auto_start** en el "php.ini".

Observación: Si no existiera, recordemos nuevamente que hay que crear una carpeta “tmp” e indicar en php.ini el path hasta ella en *session.save_path.*

Ejercicio de funcionamiento

Vamos a ver las directrices para crear un sistema de autenticación de usuario para un sitio Web y restringir el acceso a las zonas que se consideren privadas.

El sistema consta de tres páginas, una para realizar la entrada introduciendo usuario y contraseña, una segunda para realizar la comprobación de los datos y una tercera que controlará durante la navegación. Además de las páginas privadas haremos una última para cerrar la sesión de una forma segura.

La primera página a la que llamaremos login.php es un formulario que solicita el usuario y la contraseña, una vez se pulse el botón de envío, **se enviarán los datos por método POST a otra página llamada conexión.php.**

En la página conexion.php se realizará una conexión con la base de datos y se comprobarán en la tabla usuarios que existe un usuario con ese nombre y clave. **Crear una base de datos llamada bdi con una tabla denominada usuarios con usuario y clave como campos.**

Una vez realizada la comprobación de que hay resultados se inicia una nueva sesión con **session_start()**; y a continuación con **sessión_register()** se le pasa el valor **autentificado** que será el nombre de dicha sesión, y que es una variable global.

Para hacer la página **login,php** incluimos un ejemplo:

```
<html>
<head>
<title></title>
</head>
<body>
<form  action="conexion.php" method="post"> <table align="center" >
<caption>Acceso al sistema</caption>
<tr> <td colspan="2">
<span>Usuario:</span><br/>
<input class="cajas" type="text" name="user"> </td> </tr>
<tr> <td colspan="2">
<span>Clave:</span><br/>
<input class="cajas" type="password" name="key"> </td> </tr>
<tr>
<td> <br/> <input type="submit"  value="Enviar"> </td>
<td> <br/> <input type="reset" value="Borrar"> </td>
</tr>
</table>
</form>
</body>
</html>
```

Si el usuario y contraseña existen se redirigirá con header a la página privada donde se quiere enviar al usuario, por ejemplo zona.php. Por lo contrario, si ese usuario no existiera en el sistema, simplemente se le enviará a la página de login, para hacer la página de conexion.php propondremos lo siguiente:

conexion.php

```
<?php
// Conexión con la base de datos
$link= mysqli_connect("localhost","root","","bdi");
$usu=$_POST['user'];
$pwd=$_POST['key'];
$sql = "select * from usuarios where usuario='$usu' and clave='$pwd'";
$resultado = mysqli_query($link,$sql);
if (mysqli_num_rows($resultado)!=0)
{
session_start();
$_SESSION['nombre']=$usu;
$_SESSION['autentificado'] = "OK";
header ("Location: zona.php");
}
else {
header ("Location: login.php");
}
mysql_free_result($resultado);
mysql_close($link);
?>
```

zona.php

```
<?php include "seguridad.php";
$nombreusuario= $_SESSION['nombre'];
?>
<h3>Bievenido <?php echo $nombreusuario ?>
</h3><br/><br/> <a href="salir.php">cerrar session</a> </div>
```

En todas las páginas privadas de la Web, debería contener un fichero que compruebe si hay una sesión activa para que pueda ser visualizada. Para ello crearemos una página .php que se llamará **seguridad.php**.
Se recoge el valor y si es diferente a OK, entonces se considerará que esa entrada no es válida pues no se ha iniciado la sesión y se le envía a la página de entrada.

```
<?php
//Inicio la sesión
session_start();
//COMPRUEBA QUE EL USUARIO ESTA AUTENTIFICADO
if ($_SESSION['autentificado'] != "OK") {
//si no existe, envío a la página de autentificación
 header("Location: login.php");
//ademas salgo de este script
exit();
}
?>
```

Por último, la página de salida, a la que llamaremos **salir.php** en la que se destruye la sesión.

```
<?php
// continuamos con la sesión
session_start();
$_SESSION = array();
session_destroy();
?>
<html>
<head><title>SALIR</title></head>
<body>
Gracias por su visita <br><br>
<a href="login.php"> Entrar de nuevo en la zona privada</a>
</body>
</html>
```

Todas las páginas que componen la zona privada deben incluir el archivo **seguridad.php**. **include (seguridad.php)** y deben terminar
con una opción salir que enlace con la página **salir.php** para finalizar de forma segura la sesión iniciada.