Cookies y sesiones

Veremos ahora dos mecanismos para intercambiar variables entre cliente y servidor: las cookies y las sesiones.

1. Cookies

Las cookies son básicamente un texto que se intercambian navegador y servidor entre distintas peticiones, con información acerca de las mismas y de lo que se ha estado haciendo. Al cerrar el navegador o la conexión, éste se guarda en ficheros locales dicha información, para poder seguir compartiéndola en futuras visitas. Sin embargo, las cookies tienen una fecha de caducidad preestablecida (dependiendo de la cookie), y pasado ese tiempo (salvo que se renueve con una nueva visita a la página), caducan y desaparecen.

La principal utilidad de las cookies es recordar a los usuarios que acceden a una web, y las cosas que hacen, pero esta capacidad también sirve para que otros rastreen esa actividad y puedan interceptar información delicada (como usuarios y contraseñas). Por ello, actualmente los navegadores permiten desactivar y/o borrar las cookies, y los sitios web que las utilizan están obligados a avisar a los usuarios de ello. Como programadores web, las cookies deben usarse para almacenar información no relevante del usuario (preferencias del sitio, lugares favoritos o, como mucho, el login para que no lo tenga que volver a escribir, pero no el password).

Hay que tener en cuenta, además, ciertos aspectos relacionados con las cookies:

- Son independientes para cada servidor, es decir, el navegador almacena y gestiona de forma separada las cookies de cada sitio web que visitamos, y unas no tienen nada que ver con otras.
- No se pueden transmitir virus, ni acceder al disco duro, desde las cookies. Es el navegador quien controla sus propios ficheros de cookies y quien los gestiona.

Ahora que ya sabemos un poco más sobre las cookies, veamos cómo crearlas y mantenerlas en PHP. Para crear una cookie se empleará la función `setcookie`, a la que le pasaremos como parámetros el nombre de la cookie, su valor, y el tiempo en que expirará, desde ahora. Este último dato suele utilizar la función time() para contar el tiempo desde ahora, y añadirle los segundos que queramos. Si ponemos 0 o no ponemos nada, durará hasta que el usuario termine su sesión (cierre sesión, o cierre el navegador). Si ponemos un valor negativo o pasado, la cookie se borrará. Esto se usa para borrar a mano las cookies y no esperar a que lo haga el navegador.

Veamos un ejemplo donde creamos una cookie llamada *colorFavorito* con un color hexadecimal que luego podríamos usar, por ejemplo, para establecer ese color de fondo para el usuario. Asignaremos a la cookie una vida de 1 hora (3600 segundos):

```
setcookie("colorFavorito", "#432FA1", time()+3600);
```

Si más adelante queremos consultar esta cookie (para extraer el color y poderlo poner como estilo, por ejemplo), usaremos el array predefinido `$_COOKIE`, pasándole entre corchetes el nombre de la cookie, y obtendremos el valor asociado a la misma.

```
$color = $_COOKIE["colorFavorito"];
echo "<style>body { background-color: " . $color . "; }</style>";
```

Ejercicio 1:

Crea una carpeta **ejercicios6** para los ejercicios de esta sesión, y dentro una subcarpeta **preferencias** con las siguientes páginas PHP:

- **preferencias.php** con un formulario que le pida al usuario su nombre y su color favorito (a elegir de una lista desplegable, o con el control input type="color" de HTML5). El formulario deberá enviarse a *guarda_prefs.php*.
- **guarda_prefs.php** que recogerá los datos del formulario anterior, y los guardará en dos cookies llamadas *nombreusu* y *colorusu*, con 5 minutos de duración. Esta página redirigirá (con *header*) a la página *index.php* una vez almacenadas las cookies.
- **index.php** que, si hay cookies guardadas del usuario, mostrará una página con el color de fondo preferido y un mensaje personalizado de bienvenida (por ejemplo, "Bienvenido, Nacho"). Si no hay cookies guardadas, mostrará una página con fondo blanco y el mensaje "Página de inicio". En cualquiera de los dos casos, habrá un enlace para ir al formulario preferencias.php.
- Opcionalmente, añade una página **borrar_prefs.php** que elimine las preferencias (cookies) del usuario actual, y redirija a *index.php*. Pon un enlace "Borrar preferencias" en el *index.php* que llame a esta página, pero que se muestre sólo cuando el usuario actual tenga preferencias guardadas.

2. Sesiones

Las sesiones son otro mecanismo de intercambio de información entre navegador y servidor, cuyo tiempo de vida es igual a toda la visita de un usuario a una web (en cuanto el usuario cierra sesión, desconecta o cierra el navegador, se pierde la información). Al igual que ocurre con las cookies, las sesiones que tengamos en dos sitios web diferentes son independientes entre sí, al igual que las sesiones que tengan dos usuarios diferentes en un mismo sitio web.

Las principales diferencias entre las sesiones y las cookies son que las sesiones no se almacenan en ficheros en el disco, sino que se guarda un registro de ellas en el servidor. Cada cliente accede a su propia sesión a través de un identificador único, una clave alfanumérica que se pasan cliente y servidor en las peticiones. Además, podemos almacenar en sesión cualquier información (no sólo textos), y no se pueden desactivar desde el navegador.

Para manejar datos en una sesión, primero deberemos crearla con la función `session_start()`. Esta función crea la sesión si no estaba creada, y en caso de que ya exista, utiliza esa sesión. Es aconsejable poner esta función al principio de todas las páginas que necesiten sesiones.

Después, utilizaremos el array `$_SESSION` para guardar y recuperar datos de la sesión. Por ejemplo, si queremos guardar el login del usuario que nos ha enviado por un formulario, pondríamos algo como:

```
$_SESSION["loginUsuario"] = $_REQUEST["login"];
```

Si queremos eliminar algún dato de la sesión, usaremos la función `unset` con dicho dato del array:

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

Si queremos cerrar la sesión entera (y perder todo lo que haya en el array `$_SESSION`), usamos la función `session_destroy()`. Es aconsejable borrar antes a mano todas las variables que hayamos creado en la sesión, para que no se queden ocupando memoria. Para ello, en lugar de hacer *unset* con cada variable, también podemos hacer:

```
$_SESSION = array();
session_destroy();
```

2.1. Ejemplo

Veamos un ejemplo un poco más completo de todo esto. Imaginemos que queremos guardarnos en sesión el e-mail de un usuario que nos llega a través de un formulario. En la página donde recogemos los datos de ese formulario haríamos algo así:

```
// Esto nos permitirá acceder al array $_SESSION
session_start();

// Comprobamos si llega un e-mail en la petición
if (isset($_REQUEST['email']))
{
    // Almacenamos el e-mail en una casilla del array $_SESSION,
    // con el nombre que queramos (por ejemplo, 'emailUsuario')
    $_SESSION['emailUsuario'] = $_REQUEST['email'];
}

// ... Resto del código de la página
```

En las páginas donde queramos acceder a esta variable (para mostrarla en pantalla, o para enviar un e-mail a ese usuario, por ejemplo), basta con que volvamos a inicializar la sesión y consultemos el valor:

```
session_start();
if (isset($_SESSION['emailUsuario']))
{
    // Hacer lo que queramos con $_SESSION['emailUsuario']
}
else
{
    // Si no existe, podemos redirigir a otra página, por ejemplo
    header("Location:otraPagina.php");
}
```

Finalmente, cuando queramos eliminar estos datos de sesión del usuario, podemos llamar a `session_destroy`. Es conveniente también eliminar una a una las variables, o limpiar el array de sesión para liberar la memoria:

```
session_start();
if (isset($_SESSION['emailUsuario']))
    unset($_SESSION['emailUsuario']);
// También serviría esto:
// $_SESSION = array();
session_destroy();
```

Ejercicio 2:

Crea una carpeta llamada **carro** con una página llamada **carro.php** con una lista de enlaces de diferentes artículos y su precio. Por ejemplo:

- Zapatillas Nike (60 euros)
- Sudadera Domyos (15 euros)
- Pala de pádel Vairo (50 euros)
- Pelota de baloncesto Molten (20 euros)

Puedes alimentar la lista desde un array de artículos previamente definido en PHP, como este:

```
$articulos = array(
    array("id" => 1, "nombre" => "Zapatillas Nike", "precio" => 60),
    array("id" => 2, "nombre" => "Sudadera Domyos", "precio" => 15),
    array("id" => 3, "nombre" => "Pala de pádel Vairo", "precio" => 50),
    array("id" => 4, "nombre" => "Pelota de baloncesto Molten", "precio" => 20)
);
```

Cada vez que el usuario haga clic en un artículo, se enviará a la propia página, recogerá el código o *id* del artículo (que se le enviará como parámetro) y con ello buscará el artículo en el catálogo y acumulará

en sesión el total de lo que ha ido comprando, junto con un listado de los artículos que ha seleccionado, para mostrarlo todo por pantalla. Por ejemplo:

```
- Zapatillas Nike (60 euros)
- Pala de pádel Vairo (50 euros)
Total comprado: 110 euros
```

Añade un enlace a la página que sea "Vaciar carro" que limpie el carro y deje así el total a 0.

Ejercicio 3:

Crea una carpeta **login** en la carpeta de ejercicios. Dentro vamos a crear un pequeño sistema de login y uso de sesiones. Crea los siguientes archivos:

- **usuarios.txt** donde vamos a guardar varios logins y passwords de prueba, separados por ":", uno por fila, como en este ejemplo:

```
usu1:pass1
usu2:pass2
prueba:passprueba
```

- **cabecera.inc** que hará lo siguiente:
    - Comprobará si existe en sesión un atributo llamado *loginusu*. Si no existe, redirigirá a la página *login.php* que haremos después. Si existe, sacará un menú de enlaces a las páginas *pag1.php* y *pag2.php*
    - Crea las páginas **index.php**, **pag1.php** y **pag2.php** que incluyan la cabecera anterior (con *require*), y como contenido, ponles simplemente un mensaje de bienvenida diferente para cada página (para identificarlas).
    - Finalmente, la página de **login.php** tendrá un formulario para pedirle al usuario su login y password, y se llamará a sí misma. Los datos que reciba del formulario, los comparará con los del fichero *usuarios.txt* para ver si coincide con alguno. Si coincide, creará un atributo *loginusu* en sesión con el login del usuario y redirigirá a *index.php*. Si no coincide con ninguno, se volverá a cargar el propio formulario, con un mensaje de error en rojo indicando que el login o la contraseña son incorrectos.

3. Cuándo usar cookies o sesiones

Ya hemos visto estos dos mecanismos de compartir y actualizar información entre cliente y servidor, pero... ¿cuándo es conveniente utilizar cookies y cuándo sesiones? Básicamente debemos regirnos por estas sencillas reglas:

- Usaremos cookies para almacenar información simple (texto) y no relevante para el funcionamiento de la web (porque los navegadores pueden desactivar las cookies). Por ejemplo, las últimas búsquedas que ha hecho un usuario en una web, o el login del usuario para recordárselo en el formulario la próxima vez que intente loguearse (aunque el login también deberemos almacenarlo en sesiones para recordar quién ha entrado hasta que cierre sesión).
- Usaremos sesiones cuando queramos almacenar datos complejos, o cruciales para el funcionamiento de la web, ya que no pueden ser desactivadas por el navegador. Por ejemplo, el login del usuario (no el password) es útil almacenarlo en sesiones, para tener la certeza de poder identificarlo en cada página que visita (si estuviera en cookies, podrían desactivarse y perder quién ha entrado). Los elementos de un carro de la compra que tengamos añadidos de una tienda virtual, también se deberían almacenar en sesiones, pues son (o pueden ser) datos complejos, no simple texto.
- No usaremos ni una ni otra para almacenar información privada del usuario, como por ejemplo contraseña, número de tarjeta de crédito, etc.