COOKIES

¿Qué son las cookies?

Las cookies son un mecanismo que utilizan los servidores web para guardar información en el ordenador del usuario y recuperarla cada vez que el navegador les pide una página. La información se guarda en el ordenador del usuario en forma de texto y está formada por parejas (nombre de la cookie y valor de la cookie). Esta información se utiliza para numerosos fines. En el manual de PHP se ofrece un capítulo dedicado a las cookies.

La característica fundamental de utilizar cookies es el hecho de que dicha información se almacena en el ordenador cliente, es decir, es responsabilidad del navegador. Esto permite delegar dicha responsabilidad, liberando al servidor de la tarea de mantener esa información.

Sin embargo, aunque esto es una ventaja, también se corre el riesgo de que el usuario modifique dicha información o incluso que la elimine. Por ello, la **información recibida de una cookie no es segura, es necesario sanitizarla**.

Existen una serie de limitaciones en los navegadores:

- **Los servidores web sólo pueden acceder a cookies establecidas a su propio dominio**. Este dominio es establecido por el navegador cuando el servidor crea una nueva cookie, y sólo puede ser un dominio o subdominio del servidor.
- Según el **protocolo HTTP**, las cookies no pueden ser más grandes de **4096 Bytes (4KB)**.
- Hay un **límite de cookies por dominio**. Depende del navegador, pero suelen ser 20 cookies.
- También hay un **límite en el número total de cookies en el disco duro del cliente**. Suele ser de unas 300 cookies. Cuando se llega a este número, una cookie antigua se elimina antes de crear la nueva.

Desde PHP, se pueden usar las *cookies* usando la función ***setcookie()*** y el array global ***$_COOKIE***. Vamos a ver cómo.

Crear cookies

Esta función define una *cookie* que se enviará al cliente junto con el resto de las cabeceras de HTTP. Devuelve *true* si la cookie se envía con éxito o *false* en caso contrario.

Su sintaxis es:

```
setcookie(
   string $name,
   string $value = "",
```

```
    int $expires = 0,
    string $path = "",
    string $domain = "",
    bool $secure = false,
    bool $httponly = false
): bool
```

La función *setcookie()* admite varios parámetros, la mayor parte de ellos optativos:

- **name**: El nombre de la *cookie*. Este es el único obligatorio.
- **value**: El valor de la *cookie*.
- **expire**: El tiempo que la *cookie* tardará en expirar. Se trata de una fecha expresada en formato Unix. Añadiendo el tiempo de duración a time().
- **path**: La ruta del servidor para la que la *cookie* estará disponible. Si se utiliza '/', la *cookie* estará disponible en la totalidad del dominio.
- **domain**: El dominio para el cual la *cookie* está disponible.
- **secure: Si la cookie solo debería enviarse en caso de conexión https, pon este argument a true.**
- **httponly: La opción $solohttp hace que la cookie sólo esté disponible a través de http, por lo que lenguajes del lado del servidor como JavaScript no pueden acceder a la cookie. Esto ayuda a prevenir ataques XSS, aunque si se quiere acceso a las cookies desde JavaScript se tiene que activar.**

Aquí tienes tres ejemplos de envío de la misma cookie:

```
<?php
$value = "Esta es una cookie de ejemplo";

setcookie("VaderQuote", $value); La cookies expira al cerrar el navegador
setcookie("VaderQuote", $value, time()+3600);  // la cookie expira en una hora
setcookie("VaderQuote", "prueba", time()+3600, $httponly=1); //duración 1 hora y parámetro solo HTTP a true

?>
```

Es muy **importante que los nombres de las cookies no coincidan con los nombres de controles de los formularios, porque PHP puede incluir los valores de las cookies en la matriz $_REQUEST** (depende del valor de la directiva request_order en php.ini, por seguridad está desactivado de manera que el valor e la directiva es request_order="GP"). Es decir, que **si el nombre de una cookie coincide con el nombre de un control, en $_REQUEST sólo se guardará el valor de la cookie, no el del control**.

Hay que tener la precaución de **utilizar la función setcookie() antes de empezar a escribir el contenido de la página**, porque si no PHP producirá un aviso y no se creará la cookie. El motivo es que las cookies se crean mediante cabeceras de respuesta HTTP y las cabeceras se envían antes del texto de la página. Es decir, cuando PHP encuentra una instrucción que escribe texto, cierra automáticamente la cabecera; si a continuación PHP encuentra en el programa la función setcookie(), da un aviso porque ya se han enviado las cabeceras y no se crea la cookie. El ejemplo siguiente muestra código incorrecto, ya que

utiliza la función setcookie() después de haber escrito texto, y el mensaje de aviso generado por PHP.

`<?php`
`// Este código es incorrecto, la cabecera se crea después de crear texto`
`print "<p>Hola</p>";`
`setcookie("nombre", "Pepito Conejo");`
`?>`
	Hola
	
	Warning: Cannot modify header information - headers already sent by (output started at prueba.php on line 3) in prueba.php on line 4

Nota: En algunos casos este código incorrecto puede no generar un aviso y la cookie puede crearse, dependiendo de la configuración de la directiva **output_buffering**.

Utilizar cookies

Cuando el navegador solicita una página PHP a un servidor (un dominio) que ha guardado previamente cookies en ese ordenador, el navegador incluye en la cabecera de la petición HTTP todas las cookies (el nombre y el valor) creadas anteriormente por ese servidor.

El programa PHP recibe los nombres y valores de las cookies y se guardan automáticamente en la matriz $_COOKIE. Su manejo es igual que el de las variables $_POST y $_GET

El ejemplo siguiente saluda al usuario por su nombre si el nombre del usuario estaba guardado en una cookie.

```
<?php
if (isset($_COOKIE["nombre"])) {
     echo "<p>Su nombre es". htmlspecialchars($_COOKIE[nombre])."</p>";
} else {
    echo "<p>No sé su nombre.</p>"; }
?>
```

En caso de que se haya guardado una matriz en forma de cookies, el ejemplo sería el siguiente:

```
<?php
if (isset($_COOKIE["datos"]["nombre"]) && isset($_COOKIE["datos"]["apellidos"]))
{
    echo "<p>Su nombre es ". htmlspecialchars($_COOKIE["datos"]["nombre"])." ". .
htmlspecialchars($_COOKIE["datos"]["apellidos"])."</p>";
} else {
echo            sé su nombre.</p>";
}
?>
```

Un detalle importante a tener en cuenta al trabajar con cookies es el orden en que se realiza el envío y la creación de cookies, así como su disponibilidad en $_COOKIES:

- cuando una página pide al navegador que cree una cookie, el valor de la cookie no está disponible en $_COOKIE en esa página.

- el valor de la cookie estará disponible en $_COOKIE en páginas posteriores, cuando el navegador las pida y envíe el valor de la cookie en la petición.

Ejemplo

Por ello, el siguiente programa no dará el mismo resultado cuando se ejecute por primera vez que las veces posteriores:

```
<?php
setcookie("nombre", "Pepito Conejo");

if (isset($_COOKIE["nombre"]))                {
    echo   "<p>Su  nombre  es".htmlspecialchars($_COOKIE[nombre]).”</p>";
} else {
    print "<p>No sé su nombre.</p>";
}
?>
```

La primera vez que se ejecuta este programa, ocurren las siguientes cosas en este orden:

- el navegador pide la página, pero no envía con la petición el valor de ninguna cookie, porque la cookie todavía no existe
- el servidor envía la página:
  - en la cabecera de respuesta, el servidor incluye la petición de creación de la cookie.
  - en la página escribe "No sé su nombre" porque no ha recibido ninguna cookie del navegador.

La segunda vez (y las siguientes) que se ejecuta este programa, ocurren las siguientes cosas en este orden:

- el navegador pide la página y envía con la petición el valor de la cookie "nombre".
- el servidor envía la página:
  - en la cabecera de respuesta, el servidor incluye la petición de creación de la cookie (como es el mismo valor, la cookie se queda igual).
  - en la página escribe "Su nombre es Pepito Conejo" porque ha recibido la cookie del navegador.

Ejemplo almacena veces y momento de entrada anterior

```
if (empty($_COOKIE["Veces"]))
setcookie("Veces",
1); else {
setcookie("Veces", $_COOKIE["Veces"] + 1);
$Fecha_anterior = htmlspecialchars($_COOKIE[Momento][Fecha]);
$Hora_anterior = htmlspecialchars ($_COOKIE[Momento][Hora]);
}
```

Aquí vemos cómo se crea y cómo se accede a una cookie. Utilizando el array $_COOKIE hemos accedido al valor de la cookie Veces.

Comprobamos si dicho valor no existe todavía (isset), en cuyo caso, se entiende que es la primera vez que se accede a la página.

Entonces creamos la cookie utilizando la función setcookie. En el primer parámetro indicamos el nombre de la cookie (Veces) y en el segundo su valor (1 en este caso porque es la primera vez).

En el caso de que no sea la primera vez, simplemente incrementamos el valor de la cookie, para lo que volvemos a utilizar la función setcookie.

En ocasiones es interesante crear cookies con subvalores. Esto es así porque tenemos limitado el número de cookies por dominio. También tenemos limitado el espacio, sólo disponemos de 4 KB para el total de la información almacenada mediante cookies, ya que éste es el tamaño que permite la mayoría de navegadores.

Por ello, podemos reducir la información enviada al servidor creando cookies como arrays:

```
$Fecha_anterior = $_COOKIE[Momento][Fecha];
$Hora_anterior = $_COOKIE[Momento][Hora];
```

La cookie Momento tiene dos subvalores: el valor Fecha y el valor Hora. Observe cómo hemos accedido a ellos.

La misma sintaxis se utiliza para crear la cookie con subvalores.

```
$fecha = getdate(time());
$dia = $fecha["mday"] . "/" . $fecha["mon"] . "/" .$fecha["year"];
$hora = $fecha["hours"] . ":" . $fecha["minutes"] . ":".
$fecha["seconds"]; setcookie("Momento[Fecha]", $dia);
setcookie("Momento[Hora]", $hora);
```

Caducidad de las cookies

La característica más importante de una cookie es su caducidad. Al crearla podemos establecer un valor entero que indique cuánto durará.

Si no se indica ese valor en la función setcookie, la cookie se mantiene "viva" sólo en la memoria del navegador por lo que desaparece al cerrarlo.

```
<?php
if (isset($Fecha_anterior))
{
        echo "Usted visitó esta página por última vez el
        <B>$Fecha_anterior</B> a las <B>$Hora_anterior</B>";
      echo "<BR>Ha visitado esta página un total de: <B>" .
      htmlspecialchars($_COOKIE["Veces"]) . "</B> veces.";
}else{
        echo "Bienvenido a nuestra página web.";
        }
?>
```

Aquí utilizamos la función **isset** para saber si la variable **$Fecha_anterior** tiene valor, en ese caso, se imprime en pantalla la fecha y hora de la última visita del usuario, además del número de veces que la ha visitado.

Tal como tenemos ahora la página, obtendríamos la primera vez el mensaje de bienvenida y, a partir de la segunda vez que se accede a la página, se nos indicaría la fecha y hora del último acceso, además del número de veces realizado.

A este último respecto, es un error habitual pensar que podemos acceder al valor actualizado de un cookie la misma vez que se modifica.

Esto no es así, podremos acceder al valor que establecemos en nuestro código php de una cookie la siguiente vez que la página se muestre en el navegador. Sin embargo, como se trata de cookies temporales (almacenadas en la memoria del navegador), al cerrar el navegador se perderán.

Para establecer la caducidad de una cookie, debemos indicar la marca de tiempo correspondiente a esa fecha. Esto lo haremos como un parámetro más de la función **setcookie**.

```
setcookie("Veces", 1, time() + (3600 * 24 * 7));
```

Con esta expresión indicamos que la cookie tiene un período de vigencia de 7 días, ya que:

- **time( )** devuelve el momento o marca horaria actual, **(3600 * 24 * 7)** representa los segundos que hay en 7 días.

Al establecer la caducidad de las cookies, éstas se almacenarán como un archivo de texto en el ordenador del usuario y de allí se obtendrán mientras no hayan caducado.

A continuación se proporciona el código completo del archivo **cookies.php**, este ejemplo utiliza dos cookies, una de ellas almacena el número de veces que el usuario entra en la página y la otra almacena dos valores (array), la fecha y la hora de la última entrada.

```php
<?php
// Guardamos la fecha actual para guardarla
$fecha = getdate(time());
$dia = $fecha["mday"] . "/" . $fecha["mon"] . "/" . $fecha["year"];
$hora = $fecha["hours"] . ":" . $fecha["minutes"] . ":" . $fecha["seconds"];
/*
*     Si la cookie "Veces"no existe la creamos por primera vez inicializandola a 1
*     Guardamos tambien dia y hora actual de conexión
*     Para el ejemplo ponemos duración mientras esté abierto el navegador (sin tiempo)
* La cookie Momento la creamos como un array donde almacenaremos día y hora
 */
if (! isset($_COOKIE["Veces"])) {
setcookie("Veces", 1);
    setcookie("Momento[Fecha]", $dia);
setcookie("Momento[Hora]", $hora);
} else {
    /*
*     Si ya existe previamente se incrementa en uno y se actualiza el momento de
conexión
*     Vemos que para actualizar el valor de la cookie se hace tambien con setcookie
pasándole el nuevo valor
*          */
    setcookie("Veces", $_COOKIE["Veces"] + 1);
setcookie("Momento[Fecha]", $dia);     setcookie("Momento[Hora]",
$hora);

    /*
*     Aunque ya hemos modificado con setcookie no será visible en el servidor hasta la
siguiente ejecución de la página.
     */

    echo "Te has conectado ". htmlspecialchars($_COOKIE["Veces"]). " veces. La última
vez fué el día ". htmlspecialchars($_COOKIE["Momento"]["Fecha"])." a las ".
htmlspecialchars($_COOKIE["Momento"]["Hora"]); } ?>
```

Modificar cookies

Para modificar una cookie ya existente, simplemente se debe volver a crear la cookie con el nuevo valor.

Si se quiere modificar la **fecha de expiración**, **path** o **dominio**, se ha de sobreescribir la cookie con **setcookie()** con el mismo nombre que la cookie original. Si se establece una fecha de expiración en el pasado (por ejemplo *time()-30*60*), la cookie se eliminará.

Borrar cookies

Para borrar una cookie, simplemente se debe volver a crear la cookie con un tiempo de expiración anterior al presente.

```
<?php
setcookie("nombre", "Pepito Conejo", time()-60);  // Esta cookie se borrará
inmediatamente. ?>
```

Si solamente queremos borrar el valor almacenado en la cookie sin borrar la propia cookie, simplemente se debe volver a crear la cookie, sin indicarle el valor a almacenar:

```
<?php
setcookie("nombre");  // Esta cookie no se borra, pero no guardará ningún valor.
?>
```

Dependencia del navegador

Las cookies no son la mejor forma de almacenar información de estado, principalmente porque se hace en el ordenador del usuario, donde no podemos tener el control necesario. Sin embargo, este método se utiliza por su sencillez.

Uno de los problemas que presentan es la dependencia del navegador.

Así, las cookies que se almacenan en memoria sólo están disponibles para la instancia del navegador en la que se han creado. En el caso de las cookies guardadas en disco duro, todas las instancias del navegador pueden acceder a ellas.

Como existe la posibilidad de que las cookies se almacenen en el disco duro del usuario, el navegador permite que sea el usuario quien decida si esto es posible o no.