PHP 3.4

COOKIES Y SESIONES

1.- COOKIES

A medida que las aplicaciones informáticas han ido creciendo, se han hecho necesarios métodos para mantener el estado entre diferentes visitas en una página, para mantener el valor de variables o para tener un cierto control sobre los usuarios que navegan por los servidores.

Los cookies resuelven un grave problema del protocolo HTTP: al ser un protocolo de comunicación "sin estado" (**stateless**), no es capaz de mantener información persistente entre diferentes peticiones.

Gracias a los cookies se puede compartir información entre diferentes páginas de un sitio web o hasta y todo en la misma página web pero en diferentes instantes de tiempo

Los **cookies** son sencillos archivos de texto con los cuales un sitio web almacena información en el ordenador del usuario para su consulta en posteriores visitas.

Esta información puede ser:

- cuándo fue la última **vez** que se visitó la página por un cliente,
- cuántas **veces** ha visitado ese cliente la página
- cuál fue **la dirección** de correo que facilitó para su suscripción.

Funcionamiento de una cookie

1. Cuando en la barra de dirección de nuestro navegador escribimos la URL del sitio web de destino, se intenta localizar y, si se encuentra, se envía una petición al servidor web que alberga el sitio.

2. Si el **servidor web acepta** esta petición, el navegador web comprueba que en el lado del cliente existe una *cookie* del sitio web dado.

3. Si se ha encontrado la cookie, el **navegador** envía todos los pares ***nombre-valor*** al servidor en la cabecera HTTP. Además de esos pares, se envía, la información:

- La ***fecha*** **de expiración** indica la fecha y la hora en que la cookie se considerará no válida, es decir, dejará de tener vigencia

- La ***ruta*** facilitada al servidor web para asociar varios valores en la cookie en diferentes páginas del lugar al cual se está accediendo. Cuando el servidor recibe esta información la utilizará para su uso interno y otras actividades, como la validación de usuario .

4. Si la *cookie* no es encuentra en el disco duro local, se notifica su ausencia al servidor . En este caso, el servidor genera un nuevo ID para el cliente que pidió una conexión y envía la *cookie* con esta información al navegador web del sol·licitando. El navegador entonces lo almacena en el disco duro del ordenador **cliente**.

2. LOS COOKIES Y LOS NAVEGADORES

En función del S .O. encontraremos los *cookies* en diferentes ubicaciones.

Ejemplo:

En Windows se almacenan en los carpeta ocultas:

- *C:\Users\nombre_usuario\AppData\Roaming\Microsoft\Windows\Cookies*
- *C:\Users\nombre_usuario\AppData\Roaming\Mozilla\Firefox/profilas*

Podemos acceder a ellas también dede dentro de la aplicación cliente que tengamos: *Mozilla Firefox*, etc

3. COOKIES EN PHP

Los **cookies** son un mecanismo por el cual se almacenan datos en el navegador remoto para monitorizar o identificar a los usuarios que vuelven al situado web.
PHP incluye apoyo para cookies.Los cookies se consideran como variables **globales** y pueden accederse desde cualquier parte en el *script* php. También se acceden y leen como variables normales.
http://php.net/manual/es/features.cookies.php

Creación de cookies.

En PHP la creación de cookies se hace con la función ***setcookie().***

http://php.net/manual/es/function.setcookie.php

Sintaxis:

```
bool setcookie(nomcookie,[valorcookie],
[fecha_caduca],[path],
[dominio],[seguridad])
```

Todos estos parámetros **son opcionales excepto** el **nombre**.

dónde:

- ***nomcookie****:*

Indica el nombre de la variable que tendrá el valor correspondiente en el par **nombre-valor**. Es un string

- ***valorcookie***:

tien el valor de la variable especificada por el parámetro *nombrecookie*

- ***fecha_caduca:***

La fecha de caducidad es medida en segundos y si no se especifica nada, dura la sesión.
Es un entero.

- ***path****:*

especifica la jerarquía de archivos y directorios del servidor web para considerar la cookie como válida . Por defecto vale "/" y es el valor tomad, si no especificamos nada.
Es un string.

- ***dominio****:*

Indica el servidor o nombre de dominio en el cual la cookie tiene validez. El valor por defecto es el servidor web que envió la cookie.
Es un string.

- ***seguridad****:*

Indica que se tiene que transmitir la cookie por un canal seguro. Si no indicamos nada se enviará por un canal inseguro. Si vale "1" es transmitirá usando HTTPS.
Es un entero.

Ejemplo ***cookies_crea.php:***

```php
<?php
/* HAY QUE CREAR LOS COOKIES ANTES DE CUALQUIER ETIQUETA HTML*/
$segundos_un_anyo=365 * 24 * 60 * 60;
$fecha_expiracion=time()+$segundos_un_anyo; // Caduca en un año
if(isset($_COOKIE['contador']))
      // si  existe la vble cookie, se incrementa su valor en 1
      // // y se le fija fecha de expiración
     {
        $veces=$_COOKIE['contador'] + 1;
        setcookie('contador', $veces, $fecha_expiracion);
        $mensaje = 'Número de visitas a la web: ' . $_COOKIE['contador'];
        echo $mensaje.'<br>';
      }
  else
      { // si no existe la vble cookie, se crea y se da bienvenida
        $veces=1;
        setcookie('contador', $veces, $fecha_expiracion);
        $mensaje = 'Has visitado la página por primera vez';
        echo $mensaje.'<br>';
      }
?>


<html>
<head>
      <meta charset="utf-8" />
<title> Ejemplo de cookies</title>
</head>
<body>
<h4>Ejemplo de COOKIES en página web</h4>

</body>
</html>
```

Ejecución:

Ejemplo de COOKIES en página web

Has visitado la página por primera vez

Cada vez que es recargo, se incrementa el contenido del contador de accesos

Ejemplo de COOKIES en página web

Número de visitas a la web: 1

Acceso a cookies.

Como ya se ha visto, para recuperar el valor de una cookie, se emplea el array predefinido **$_COOKIE** con el **nombre** de la cookie como índice.

`$_COOKIE['nombre_cookie'].`

http://php.net/manual/es/reserved.variables.cookies.php

No podemos leer la cookie colocada en el ordenador de nuestro visitante hasta que no recarga la página.

Ejemplo:

En el ejemplo anterior, podemos ver cómo acceder a la variable cookie ***contador***.

Si la petición de HTTP contiene información sobre cookies, PHP colocará automáticamente estas cookies como variables globales.

- Para almacenar **múltiples** valores en una cookie, trataremos a los cookies como **vectores**:

```
setcookie("valor[0]",$valor0);
setcookie("valor[1]",$valor1);
```

Donde creamos los cookies

Hace falta tener en cuenta que las cookies tienen que enviarse **antes** de mandar cualquier otra **cabecera**, esto es:

- ***Hace falta crearlas antes de cualquier etiqueta <HTML> o <head>***

Ámbito de los cookies

También es útil a veces **limitar** el ámbito **de una cookie**.

Es posible que en el nuestro servidor haya aplicaciones de diferentes empresas y no es interesante que puedan acceder a los cookies respectivas. Para esto, después de la caducidad, pueden especificarse tanto la vía de acceso a la cookie como el **dominio** en que son válidas:

```
setcookie("veces",$veces, time()+3600,"/aplicación","servidor.com");
```

Si la aplicación necesitara enviar información confidencial, podemos incluir un **último parámetro** que, cuando es **diferente a cero**, significa que la cookie no se enviará salvo que pueda hacerse mediante el uso del protocolo seguro HTTPS.

Como borrar los cookies

Si queremos **eliminar la cookie**, solo tenemos que volverla a enviar con el mismo nombre, pero con la fecha de caducidad **pasada**, o solo con el nombre de la variable a eliminar como parámetro .
Ejemplo:
cookies_borra.php:

```php
/* HAY QUE CREAR LOS COOKIES ANTES DE CUALQUIER ETIQUETA HTML*/
function borra_cookie($nombre_cookie)
{
   if (isset($_COOKIE[$nombre_cookie]))
   {
   $tiempo_vida=-1;
   //setcookie($nombre_cookie, null, $tiempo_vida, '/');// asigno valor pasado
   setcookie($nombre_cookie); //así tb se vacía un cookie
   unset($_COOKIE[$nombre_cookie]);// eliminarla del servidor
   return true;
   }
   else {
       return false;
   }
}
```

```php
// PROGRAMA PRINCIPAL
$nombre_cookie='contador';
$segundos=1000;
$valor=0;
echo "<br>Intentamos borrar manualmente el cookie.....<br>";
$borra=borra_cookie($nombre_cookie);
if ($borra) // se ha podido borrar
    {
        echo "Borrado el cookie <b>$nombre_cookie</b><br>";
        echo "Intentamos ver su contenido..... <br><br>";
        if (isset($_COOKIE[$nombre_cookie]))
            echo 'valor del cookie: '.$_COOKIE[$nombre_cookie];
        else
            echo 'No tiene contenido<br>';
    }
else    // no se ha borrado
    echo "No se ha podido borrar el cookie<br>";
```

Ejecución:

Intentamos borrar manualmente el cookie.....
Borrado el cookie **contador**
Intentamos ver su contenido.....

No tiene contenido

Ejemplo de COOKIES en página web

4.- SESIONES

Se verá en un con más detalle en un tema posterior, una vez estudiada la forma de accesos a bases de datos Mysql con PHP

El uso de sesiones es un método ampliamente extendido en cualquier aplicación de una cierta entidad.

Básicamente una **sesión** alcanza la **secuencia de páginas** que un usuario visita en un sitio web , desde que entra en el nuestro lugar, hasta que lo abandona .

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

Ejemplos de esto podrían ser tanto la banca en línea, en la cual cada usuario dispone de un área de navegación personalizada, como el típico *carro de la compra*, en el cual 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.

Dado que los comunicaciones que se establecen a través del protocolo HTTP son independientes unas de otras , cerrando 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 de esa información compartida, al margen del funcionamiento de este tipo de protocolo .

Una sesión, 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 tiene que diferenciarse y ser ***independiente*** del resto (puesto que una página puede ser accedida al mismo tiempo por varios usuarios), por la que PHP asigna un ***identificador*** por cada sesión que se inicie, siendo este identificador una secuencia alfanumérica generada ***automáticamente***.

El proceso en cualquier lenguaje de programación podría ser una cosa así:

- Existe una sesión ?
- Si existe la retomamos
- Si no existe creamos una nueva
- Generar un identificador único

5.- MANEJO DE SESIONES

Funciones de PHP para el manejo de sesiones (cuando **register_globals ON)**

Cuando se pone la variable **register_globals=*ON*** se tiene acceso a todas los variables globales del servidor.

Esta variable es encuentra en el archivo de configuración ***c:\xamp\php.ini.*** Por defecto suele estar a ***OFF***.

- **`session_start()`**

Inicializa una sesión y le asigna un identificador ***único***.

Si la sesión ya está iniciada, ***carga*** todas los variables de sesión.

- **`session_register(variable)`**

Registra una variable de sesión.

- **`session_unregister(variable)`**

Elimina una variable de sesión.

- **`session_is_registered(variable)`**

Comprueba si una variable está registrada. Devuelve *true* en caso afirmativo y *false* en caso contrario

- **`session_destroy()`**

Cierra una sesión

Funciones de PHP para el manejo de sesiones (con `register_globales OFF`):

- **`session_start()`**

Inicializa una sesión y le asigna un ***identificador*** de sesión ***único***. Si la sesión ya está iniciada, **carga todas** los variables de sesión.

- **`$_SESSION['nombre'] = valor;`**

Registra una variable de sesión.

- **`unset($_SESSION['nombre']);`**

Elimina una variable de sesión.

- **`isset($_SESSION['nombre'])`**

Comprueba si una variable está registrada. Devuelve *true* en caso afirmativo y *false* en caso contrario

- **`session_destroy()`**

Cierra una sesión

Procedimiento manejo de sesiones.

- **Todas las páginas** tienen que realizar un llamamiento a ***session_start()*** para cargar las variables de la sesión
- Este llamamiento tiene que estar colocado antes de cualquier código HTML
- Conviene invocar a ***session_destroy()*** para cerrar la sesión
- Para cada una de las páginas, el intérprete de PHP comprueba si existe la sesión. En caso afirmativo, se retoma, si no, se crea una nueva (generando un nuevo identificador).

Iniciar/prolongar una sesión

Existen **3** maneras:

- invocando a la función **session_start()**
- a través de la función **session_register()** (creación de variables).
- activando la directiva **session.auto_start** en el "*php.ini".*

Si no existiera, recordamos nuevamente que hace falta crear una carpeta “tmp” e indicar en ***php.ini*** el path hasta ella en ***session.save_path****.*

Otras funciones de interés :

- **session_unset()**: " *vacía*" el contenido de los variables de sesión .
- **session_name([string nombre])**:

Tiene 2 comportamientos:

- si se invoca sin parámetro devuelve el nombre de la variable interna que conté el identificador de sesión (por defecto se asigna el valor de la directiva *session.name* de *php.ini*).
- si existe parámetro, es cambia el contenido de esta variable, pero será necesario hacerlo antes del comienzo de la sesión en cada script para prolongar el suyo efecto.

- **session_save_path([string path])**:

Devuelve el camino al directorio donde se sitúan los ficheros asociados a la sesión. Es modificable, introduciendo una nueva ruta.

Para prolongar su efecto, es necesario incluir esta función en **cada script** que haga uso de los sesiones.