U.T. 5_1: Cookies y Sesiones.

Cookies y Sesiones: Cookies

- El protocolo HTTP es un protocolo sin estado.
  - Cada vez que solicitamos una página a un servidor representa una conexión distinta.
- En una aplicación web es necesario casi siempre mantener el estado de la sesión, es decir, mantener algo que nos permita vincular una petición con otra
- Las cookies (galletas informáticas) son simples ficheros de texto mediante los cuales un sitio web almacena información en el ordenador del usuario.
- Solamente el sitio que ha creado la cookie es capaz de volver a leerla.
- Si se envía solamente el nombre, la cookie será eliminada en el cliente.
- Ojo! El **usuario** puede **deshabilitar** las cookies en el navegador

Cookies y Sesiones: Cookies

- El manejo de las cookies en PHP es extremadamente sencillo:
  - En el 1er paso se **envía** la cookie:
    - llamar a la función **setcookie** para crear la cookie en el cliente
  - En las posteriores peticiones que recibamos de ese cliente vendrá incrustada la cookie => se **consulta** la información almacenada en ella
    - acceder mediante la superglobal **$_COOKIE**.

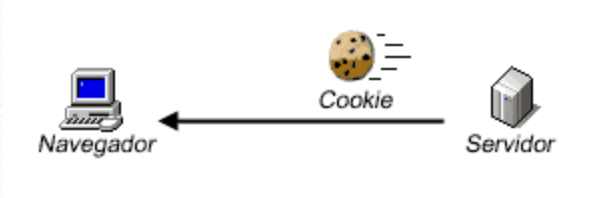
- Las cookies se envían en las cabeceras de las transacciones HTTP

⇒deben enviarse antes que cualquier otra cabecera HTML

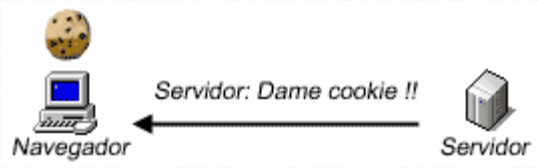
(restricción de las cookies no de php)

Cookies y Sesiones: Cookies

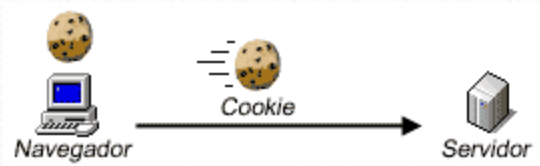
- Después de crear una cookie, cada vez que el navegador visita el sitio, envía dicha cookie.
  - La cookie es **enviada al navegador** desde el servidor y si éste la acepta permanece en él.



  - Las **páginas piden la cookie** al navegador...



  - El **navegador envía la cookie**, permitiendo la identificación del usuario por parte del servidor.



Cookies y Sesiones: Cookies

- Una cookie es un bloque de texto constituido por varios campos. Todos son opcionales excepto el nombre de la cookie.
  - **Nombre:** Nombre de la cookie
  - **Valor**
    - Valor asociado a la cookie (se envía empleando la codificación URL)
  - **fecha expiración:** Fecha de expiración de la cookie.
  - **Path**
    - Subconjunto de URLs para los que la cookie es válida
  - **Dominio**
    - Rango de dominios para los que la cookie es válida en el servidor
  - **Segura**
    - Indica si la cookie se debe transmitir exclusivamente sobre https
  - **Httponly:** Accesible sólo por el protocolo HTTP (no script JavaScript)
- Tamaño máximo 4Kb

Cookies y Sesiones: Cookies

- **Crear cookie**

```
setcookie($nombre,$valor,$fecha_expiración,$path_valido,$dominio,$segura,
                                $httponly)
```

- Ejemplos
  - $fecha_expiracion=time()+60*60*24*365; → dentro de 1 año en segundos
  - $fecha_expiracion=mktime(0,0,01,1,2015); → 1 Enero 2015
  - $path_valido=$_SERVER['REQUEST_URI']; → Sólo el php actual
  - $path_valido="/"; → todo el sitio
  - $nombre=preferencias[idioma] -> recibimos los datos de un array

Cookies y Sesiones: Cookies

- **Borrar cookie**
  - Llamar a la función setcookie con una fecha anterior a la actual.

    setcookie("mail","",time()-1000,"/");
  - Enviar una cookie sólo con nombre

    setcookie("mail");
- **Cookie de sesión**
  - Sólo existe mientras no cerremos la ventana del navegador.
  - Poner como **fecha de expiración de la cookie**, el valor **cero**.

    setcookie(,,0);

Cookies y Sesiones: Cookies

- **EJERCICIO**

  Crear varias cookies desde un script PHP que luego recibiremos desde otro.

  Crea ***setcookies.php*** con el siguiente contenido:

```
<?php
    setcookie('Fecha',date('Y-m-d H:i:s'));
    setcookie('preferencias[idioma]','español');
    setcookie('preferencias[fondo]','rojo');
?>
```

Cookies y Sesiones: Sesiones

- Una ***aplicación web*** es algo más que una simple consulta.
- A veces es necesario ***mantener el estado*** de una conexión entre distintas páginas o entre distintas visitas a un mismo sitio. Por ejemplo para:
  - Acumular una cesta de la compra
  - Control de acceso de los usuarios
  - Conocer los pasos de la navegación y preferencias del usuario
  - Actualizar una base de datos
  - etc.

Cookies y Sesiones: Sesiones

- HTTP es un protocolo sin estado.
- Para mantener los valores de las variables PHP a lo largo de toda la navegación (ejecución de varios scripts) usamos:
  - **Cookies:** los valores de las variables se almacenan en el cliente.
  - **Sesiones:** los valores de las variables se almacenan en el servidor

Cookies y Sesiones: Sesiones

- Las sesiones son una manera de guardar información, específica para cada usuario, durante toda su visita.
- Cada usuario que entra en un sitio abre una sesión, independiente de la sesión de otros usuarios.
- En la sesión de un usuario podemos almacenar todo tipo de datos:
  - nombre del usuario
  - productos de un hipotético carrito de la compra
  - preferencias de visualización o trabajo
  - páginas por las que ha pasado, etc.
- Todas estas informaciones se guardan en lo que denominamos **variables de sesión**.

Cookies y Sesiones: Sesiones

- **SESIÓN:** Forma que tiene php de *mantener el valor de las variables* a lo largo de toda la navegación.
  - Las cookies permiten almacenar poca información, 4kb.
  - Muchos usuarios desactivan las cookies.
  - Las sesiones en lugar de almacenar la información en el cliente la almacenan en el servidor.
  - PHP internamente genera un *identificador de sesión único*, que sirve para saber las variables de sesión que pertenecen a cada usuario.
  - Lo único que guardan las sesiones en el cliente es una ***cookie*** que contiene el ***id de la sesión.***

Cookies y Sesiones: Sesiones

- Las sesiones de php siguen funcionando aunque las cookies estén desactivadas.
  - Cuando el servidor web detecta que las cookies no están activadas, añade automáticamente el *id de sesión* como un ***query string*** en todos los enlaces de la página.
  - Para que esto funcione hay que cumplir estos requisitos:
    - Todas las páginas tienen que tener la extensión .php, pues de lo contrario php no puede añadir la id sesión a páginas que no sean .php.
    - Además es necesario que **session.use_trans_sid** esté activado en php.ini.
    - Anchors y formularios, pero no headers
    - Sólo en direcciones relativas (no absolutas).
    - htmlspecialchars(SID)

Cookies y Sesiones: Sesiones

- Configurar **php.ini** para que funcionen las sesiones:
  - Sección [SESSION]

```
session.save_handler = files
session.save_path = C:\WINDOWS\TEMP
session.use_only_cookies = 1
session.use_trans_sid = 1
```

Cookies y Sesiones: Sesiones

- Para que las sesiones funcionen, no pueden aparecer páginas estáticas (documentos HTML) en pasos intermedios.
- Para que un script forme parte de una sesión, al inicio del script y antes de generar cualquier tipo de etiqueta o contenido HTML debe ejecutar la función **session_start()**.
- Cada sesión es independiente y única para cada usuario
- El manejo de las sesiones se realiza de la siguiente forma:
  - Todas las páginas deben realizar una llamada a *session_start()* para cargar las variables de la sesión
  - Esta llamada debe estar colocada antes de cualquier código HTML
  - Conviene llamar a *session_destroy()* para cerrar la sesión

Cookies y Sesiones: Sesiones

- Funciones de PHP para el manejo de sesiones
  - **session_start ()**
    - Inicializa una sesión y le asigna un identificador de sesión único. Si no existe la crea y si ya existe la recupera (carga todas las variables de sesión).
  - **session_destroy ()**
    - Cierra una sesión
    - No destruye ninguna de las variables globales asociadas con la sesión, ni destruye la cookie de sesión.
    - Para destruir la sesión completamente, como desconectar al usuario, el id de sesión también debe ser destruido, es decir, se debe borrar la cookie de sesión.
  - **session_name() :** Devuelve el nombre de la sesión.
  - **session_id()** : Devuelve y/o establece el identificador de la sesión.

Cookies y Sesiones: Sesiones

- Funciones de PHP para el manejo de sesiones:
  - Registra o modifica una variable de sesión

    $_SESSION['nombre'] = valor;
  - Elimina una o todas las variables de sesión

    unset ($_SESSION['nombre']);

    $_SESSION=array();
  - Comprueba si una variable está registrada

    if (isset($_SESSION['nombre']))

Cookies y Sesiones: Sesiones

- Funciones de PHP para el manejo de sesiones:
  - **session_cache_expire()**: Devuelve/asigna la caducidad de la caché actual.
  - **session_cache_limiter()**: Obtener y/o establecer el limitador de caché actual.
  - **session_commit()**: alias de session_write_close.
  - **session_decode()**: decodifica la información de sesión desde una cadena.
  - **session_encode()**: codifica la información de la sesión actual y los devuelve como una cadena.
  - **session_get_cookie_params()**: Devuelve una matriz asociativa con la información de la cookie de la sesión.

Cookies y Sesiones: Sesiones

- **Array session_get_cookie_params()**: Los índices de éste array son:
  - '**lifetime**': Caducidad de la cookie de sesión
  - '**path**': Ruta para la cookie de sesión.
  - '**domain**': Dominio del servidor que genera la cookie.
  - '**secure**': Modo de envío de la cookie
  - '**httponly**': La cookie sólo se propaga por HTTP.

```
// Caducar la cookie de sesión con los mismos parámetros de creación

   $CookieInfo = session_get_cookie_params();
   if ( (empty($CookieInfo['domain'])) && (empty($CookieInfo['secure'])) ) {
      setcookie(session_name(), '', time()-3600, $CookieInfo['path']);
   } elseif (empty($CookieInfo['secure'])) {
      setcookie(session_name(), '', time()-3600, $CookieInfo['path'], $CookieInfo['domain']);
   } else {
      setcookie(session_name(), '', time()-3600, $CookieInfo['path'], $CookieInfo['domain'],
                $CookieInfo['secure']);
   }
   session_destroy();
```

Cookies y Sesiones: Sesiones

- Funciones de PHP para el manejo de sesiones:
  - **session_is_registered()**: Averiguar si una variable global está registrada en una sesión.
  - **session_regenerate_id()**: Actualiza el id de sesión actual con un nuevo valor más reciente.
  - **session_save_path()**: Obtener y/o establecer la ruta de almacenamiento de la sesión actual.
  - **session_set_cookie_params(plifetime,ppath,pdomain,psecure,httponly)** : Establecer los parámetros de la cookie de sesión.
  - **session_write_close()**: escribir información de sesión y finalizar la sesión.

Cookies y Sesiones: Sesiones

- Contador1:

```
<?php
session_start();
$_SESSION['contador']++;
?>
<html>
<a href="contador2.php">Página que muestra el contador</a>
</html>
```

- Contador2:

```
<?php session_start(); ?>
<html><body>
<?php
echo "contador: " . $_SESSION['contador'];
?>
<br><a href="contador1.php">[ Volver ]</a>
<br><a href="contador3.php">[ Terminar]</a>
</body></html>
```

Cookies y Sesiones: Sesiones

- Contador3:

```
<?php
session_start();
Unset($_SESSION['contador']);
// $_SESSION=array();   para borrar todas las vars de sesión
Setcookie(session_name(),,time()-3600);
Session_destroy();
?>
<html>
<body>
 Sesión terminada
</body>
</html>
```

Cookies y Sesiones: Sesiones

- Autenticación de usuarios
  - Una cuestión frecuente en un sitio web es controlar el acceso de los usuarios a una zona determinada del mismo.
  - La autenticación de usuarios puede realizarse en el propio servidor web. Así, en Apache los ficheros ***.htaccess*** permiten limitar el acceso a un determinado recurso del servidor.
  - Una alternativa más compleja pero más flexible es utilizar PHP junto con una base de datos para controlar el acceso de los usuarios. Para ello se utilizan las sesiones.

Cookies y Sesiones: Sesiones

- Esquema de una página que utiliza sesiones para autenticar usuarios:

autentica.php

```
<?php       //crea la variable de sesión de usuario autenticado para consultarla después
  session_start();
  if (!isset($_SESSION["autentificado"])){
    if (isset($_POST["usuario"]) && isset($_POST["contrasena"])){
      if ($_POST["usuario"]=="carmen" && $_POST["contrasena"]=="carmen"){
                              $_SESSION["autentificado"]="SI";
                              header("Location: aplicacionsegura.php");
                }
      else  // Credenciales erróneas
                              header("Location: index.php?errorusuario=1);
    }
    else  //No ha rellenado el formulario de autenticación
                header("Location: index.php");
 }
  else  // Ya está acreditado y no ha cerrado la sesión aún
                header("Location: aplicacionsegura.php");
?>
```

Cookies y Sesiones: Sesiones

- Fichero que controla el acceso restringido en las páginas que lo incluyan: **seguridad.php**

```
<?php      // comprueba la existencia de usuario autenticado
   session_start();
   if ($_SESSION["autentificado"]!="SI"){
              header("Location: index.php");
              exit();
   }
?>
```

Cookies y Sesiones: Sesiones

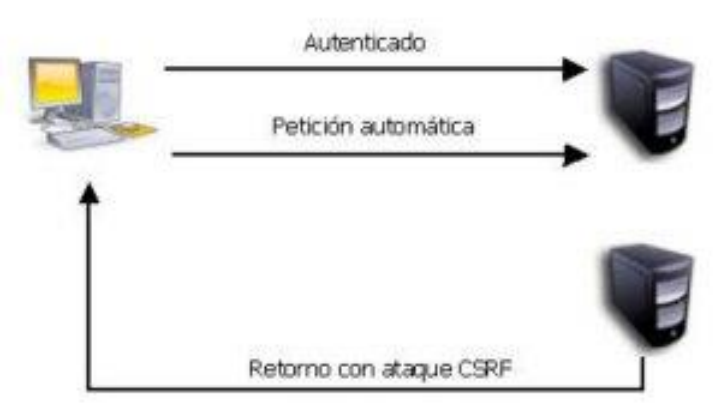
- Directivas del php.ini asociadas a la configuración de sesiones:
  - **session.auto_start**: Indica si el módulo que administra las sesiones se inicia automáticamente o no.
  - **session.cache_expire**: Tiempo de vida de las páginas de la sesión almacenadas en caché.
  - **session.cache_limiter**: Método que se aplicará para almacenar las páginas de la sesión en el caché (none, ***nocache***, private, private_no_expire, public).
  - **session.cookie_domain**: Dominio de la sesión.
  - **session.cookie_lifetime**: Duración de la cookie en segundos.
  - **session.cookie_path**: Ruta de acceso que se asigna a session_cookie.
  - **session.gc_maxlifetime**: Cantidad de segundos tras los que los datos de sesión pueden ser eliminados.

Cookies y Sesiones: Sesiones

- Directivas del php.ini asociadas a la configuración de sesiones:
  - **session.name**: Indica el nombre de la sesión que se utiliza como nombre de la cookie. (PHPSESSID).
  - **session.save_handler:** Determina el nombre del controlador que se utiliza para almacenar los datos asociados con la sesión (files).
  - **session.save_path**: Ruta de acceso donde se almacenan los datos asociados a las cookies (/tmp).
  - **session.use_cookies**: Indica si se van a usar cookies para guardar el identificador de sesión en el cliente (1).
  - **session.use_trans_sid**: Indica si la inclusión del identificador de sesión transparente está activada o no, si se utiliza enable-trans-sid (1).
  - **url_rewriter.tags**: Indica las etiquetas html que serán reescritas para propagar el id de sesión (a=href, area=href, frame=src, input=src, form=fakeentry).

Cookies y Sesiones: Tokens de formulario

- Ataques **CSRF/XSRF (*Cross-site request forgery)***.



Cookies y Sesiones: Tokens de formulario

- Evitan que se pueda **reenviar** un formulario
- Evitan ataques **CSRF/XSRF (*Cross-site request forgery)***.
- En definitiva, un **token** de un formulario es un **campo oculto** que incluye un valor único (generalmente un número generado con la función **uniqid()**) y, a la vez, almacenamos ese mismo valor en una **variable de la sesión** del usuario.
- Cuando el procesador del formulario recibe datos **comprueba si el token recibido del formulario coincide con el de la sesión.**

  => Si no coincide descarta la petición;

  => si coincide procesa la petición y **borra la variable de sesión que contenía el valor del token**.

Cookies y Sesiones: Tokens de formulario

- Realiza el ejercicio hoja_9 para provocar un ataque CSRF y realizar las modificaciones necesarias para evitarlo. Utiliza los scripts de otros ejercicios:
  - acreditacion.php
  - transferencia.php
  - procesar_transferencia.php
  - informacion.php

Y añade el script:

  - horoscopo.php

Cookies y Sesiones: Tokens de formulario

- Doc sobre ataques CSRF:

http://www.funcion13.com/2012/08/21/preven-falsificacion-peticion-sitios-cruzados-csrf/

http://www.linuxmagazine.es/issue/50/008009_Inseguridades50.pdf

- Doc sobre ataques **XSS:**

http://osl.ugr.es/descargas/OWAND11/OWAND11%20Granada%20-%20Ataques%20XSS%20en%20Aplicaciones%20Web.pdf