# Tema 07 – Control de Sesiones

## 2º DAW – Desarrollo Web Entorno Servidor

**Profesor Juan Carlos Moreno**

**CURSO 2023/2024**

# Tabla de Contenido

# 7 Control de Sesiones

## 7.1 Introducción

Funcionamiento de la web se basa en modelo estático o no conectado. El salto de una página web a otra implica la pérdida de datos y no preserva los estados de las mismas y la única forma de garantizar la permanencia de dichos estados es mediante el control o manejo de sesiones.

### 7.1.1 Control de Sesiones

El manejo de sesiones nos permite controlar que se ejecuten de manera correcta y segura los recursos o los servicios ofrecidos por la aplicación web.

El objetivo principal es guardar información específica de cada usuario mientras este navega por una aplicación web.

Cada usuario que entra en un sitio web abre una sesión que es diferente e independiente de la sesión de otros usuarios.

La sesión evita que un usuario pase de una página a otra y pierda la información o tenga el impedimento de continuar realizando alguna operación on-line en la que tiene privilegios.

Por otro lado si no existieran las sesiones el servidor no tendría control de lo que ocurre en el cliente y podría estar asignando un espacio de trabajo a un usuario no autorizado o viceversa.

### 7.1.2 Establecer una Sesión

Procesos cuando se establece una sesión:

- A través de URL (variables de sesión)
- Uso de las cookies

#### *7.1.2.1 URL o Variables de Sesión*

Se pasan las variables o los datos de un punto a otro mediante los métodos GET o POST

Cuando un cliente se identifica al servidor éste envía al cliente un código único de identificación y gestión de sesión llamado **SID**:

- Número aleatorio generado por el servidor y que se usa para gestionar la sesión
- Será almacenado en el cliente hasta que se destruya la sesión (cookies)

En el lado del servidor además del SID se almacena un conjunto de datos relacionados con las variables de sesión declaradas.

#### *7.1.2.2 Cookies*

En caso de las cookies o galletas de información en el cliente se almacena un archivo que guarda SID y otros parámetros relacionados con la expiración.

Mediante este pequeño archivo el servidor puede identificar al cliente y puede controlar la sesión garantizando la funcionalidad operativa.

Sin embargo el uso de las cookies puede estar condicionado por lo siguiente:

- Muchos navegadores web no lo soportan
- El cliente no permite su uso
- Está desactivado su uso en el navegador

Para evitar que este inconveniente condicione la funcionalidad del sistema algunos lenguajes embebidos como PHP fuerzan a utilizar ambos sistemas, de forma que en caso de no poder usar cookies el sistema siga operando mediante el control de sesión mediante URL.

## 7.2 Control de Sesiones en PHP

PHP cuenta con todo un catálogo de funciones relacionadas con el manejo y control de sesiones y de variables de sesión:

- http://php.net/manual/es/book.session.php

Una de las grandes utilidades de las sesiones es la de guardar información acerca de la visita de un usuario específico tales como nombre, usuario, correo electrónico, contraseña, etc. Información que será preservada mientras la sesión esté vigente.

### 7.2.1 Funciones de Inicio y Cierre de Sesión

- **session_start()**: **crea una sesión** o **continúa con la actual**. Hay que ponerlo en cada página donde queramos acceder a las variables de la sesión (es lo que se llama propagar la sesión). Al llamar a esta función se verifica si se ha creado el identificador de sesión **SID**, en tal caso devuelve dicho valor y en caso contrario crea uno nuevo. Además genera una cookie en el navegador del cliente llamada **PHPSESSID** que almacena el mismo **SID** que nos ha asignado el servidor. Si en otra página me encuentro con **session_start()** comprueba la **SID** del servidor con la **SID** de la cookie **PHPSESSID**, si coinciden propaga la sesión, sino, crea una nueva.
- **session_destroy()**: Destruye la sesión actual, pero no destruye las variables de la sesión ni la cookie de la sesión. Elimina el identificador de sesión **SID** del cliente.

### 7.2.2 Funciones para Identificar Sesión

- **session_id()**: se usa para obtener o establecer el SID de sesión para la sesión actual. Si se especifica **id**, reemplazará el id de sesión actual. session_id() necesita ser llamado antes de session_start() para este propósito. Dependiendo del gestor de sesión, no todos los caracteres están permitidos dentro del id de sesión
- **session_name()**: devuelve el nombre de la sesión actual. Si se da el nombre **name**, session_name() actualizará el nombre de la sesión y devolverá el nombre *antiguo* de la sesión, así, **session_name('nombre')** permite asignar un nuevo nombre a la sesión actual.

Para que estas funciones tengan efecto han de ejecutarse antes de la función **session_start()**.

Veamos el siguiente ejemplo

```
# Ejemplo 1. Crear sesión personalizada

session_id('123carlosMoreno');
session_name('SESION-UBRIQUE');
```

```
    # Inicio ahora la sesión
    session_start();

    echo 'SID: ';
    echo session_id();
```

#### 7.2.2.1 Variables de Sesión

La información relacionada con una sesión se puede almacenar además de con las **cookies** con **variables de sesión**. Las variables de sesión se diferencian del resto en que se almacenan en un espacio de memoria del servidor junto con el **SID** de la sesión correspondiente.

##### 7.2.2.1.1 Declaración de una variable de sesión:

- $_SESSION[‘nombre variable’] = valor

Hay que tener en cuenta que para poder declarar una variable de sesión previamente hay que llamar a la función **session_start()**.

**$_SESSION** es una array asociativo y el valor que se le asigne puede ser de cualquier tipo, incluso puedo contener un nuevo array de cualquiera de los tipos: asociativo, indexado, etc…

##### 7.2.2.1.2 Destruir una variable Sesión

Hay que tener en cuenta que **session_destroy()** destruye el identificador de sesión pero no vacía las variables de sesión, para ello.

- Si queremos vaciar todas las variables de sesión usaremos **session_unset()**
- Si queremos vaciar una variable de sesión concreta usaremos **unset($_SESSION[‘variable’])**
- Hay otra forma de eliminar todas las variables de sesión **$_SESSION = []** o **$_SESSION = array()**

### 7.2.3 Ejemplos

#### 7.2.3.1 Comprobación de Variable de Sesión

```
<?
      session_start();
      if (!isset($_SESSION['nombre'])) $_SESSION['nombre']="Miguel";
?>
```

Y en otra página podríamos poner

```
<?    session_start();
      if (isset($_SESSION['nombre'])) echo $_SESSION['nombre'];
?>
```

#### 7.2.3.2 Variables de Sesión tipo Array

```
<?
      session_start();
      $_SESSION["personas"][0]["nombre"]="Pepe";
      $_SESSION["personas"][0]["Edad"]=17;
      $_SESSION["personas"][0]["Peso"]=80;
```

```
      $_SESSION["personas"][1]["nombre"]="Julián";
      $_SESSION["personas"][1]["Edad"]=25;
      $_SESSION["personas"][1]["Peso"]=120;
      // y para visualizar
      echo $_SESSION["personas"][0]["nombre"];
?>
```

Las sesiones se almacenan en el servidor, en un directorio temporal. En el fichero **php.ini** en la directiva **session.save_path**

#### *7.2.3.3 Destruir una Sesión*

```
<?php
      # Si está usando session_name("miSesion")
      session_start();

      # Destruir todas las variables de sesión.
      Session_unset();

      # Destruir la información de la sesión.
      session_destroy();
?>
```

#### *7.2.3.4 Contador de Visitas*

```
<?php
      # Iniciamos la sesión o recuperamos la anterior sesión existente
      session_start();

      # Comprobamos si la variable ya existe
      if (isset($_SESSION['visitas']))
            $_SESSION['visitas']++;
      else
            $_SESSION['visitas'] = 1;
?>
```

Si en lugar del número de visitas, quisieras almacenar el instante en que se produce cada una, la variable de sesión "visitas" deberá ser un array y por tanto tendrás que cambiar el código anterior por:

```
<?php
      # Iniciamos la sesión o recuperamos la anterior sesión existente
      session_start();
      # En cada visita añadimos un valor al array "visitas"
      $_SESSION['visitas'][] = mktime();
?>
```

## 7.3 Cookies

Ya sabemos que una de las principales utilidades de las cookies (galletas) es la de solventar el problema de la falta de estado en la navegación a través de las páginas web.

Con las cookies, pequeñas porciones de información se quedan registradas en el navegador permitiendo identificar a este a través de diferentes páginas de un mismo sitio e incluso durante visitas entre distintos días.

Realmente las cookies no son más que cadenas de texto que son enviadas desde el servidor al cliente (navegador) y almacenadas en este, luego el navegador envía estas cookies al servidor permitiendo así la identificación del cliente en el servidor.

### 7.3.1 Restricciones de las cookies

- 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.

Las cookies tienen una **fecha de expiración**. Esta fecha permite al navegador eliminar cookies antiguas cuando ya no son requeridas por el servidor web. Si la fecha de expiración está vacía, la cookie se eliminará cuando finalice la conexión con el servidor. Esto ocurrirá cuando el usuario cierre la ventana o pestaña del sitio web, o directamente cierre el navegador. Estas cookies, normalmente denominadas **session cookies**, son usadas principalmente para guardar ajustes temporales.

### 7.3.2 Funcionamiento de las Cookies

Las cookies se transmiten entre el navegador y el servidor web utilizando los encabezados del protocolo HTTP. Por ello, las sentencias **setcookie** deben enviarse antes de que el navegador muestre información alguna en pantalla.

El proceso de recuperación de la información que almacena una cookie es muy simple. Cuando accedes a un sitio web, el navegador le envía de forma automática todo el contenido de las cookies que almacene relativas a ese sitio en concreto. Desde PHP puedes acceder a esta información por medio del array **$_COOKIE**.

Siempre que utilices cookies en una aplicación web, debes tener en cuenta que en última instancia su disponibilidad está controlada por el cliente. Por ejemplo, algunos usuarios deshabilitan las cookies en el navegador porque piensan que la información que almacenan puede suponer un potencial problema de seguridad. O la información que almacenan puede llegar a perderse porque el usuario decide formatear el equipo o simplemente eliminarlas de su sistema.

Si una vez almacenada una cookie en el navegador quieres eliminarla antes de que expire, puedes utilizar la misma función **setcookie** pero indicando una fecha de caducidad anterior a la actual.

### 7.3.3 Función setcookie()

El manejo de cookies en PHP se realiza mediante el uso de la función **setcookie**(), esta función está disponible a partir de la versión 3 de PHP.

Sintaxis de setcookie():

```
int setcookie (string Nombre [, string Valor [, int Expire [, string
Path [, string Dominio [, int Secure]]]]])
```

**Setcookie()** define una cookie que es enviada junto con el resto de la información de la cabecera(header). Las cookies deben ser enviadas antes de cualquier tag de html, por lo tanto deberemos realizar la llamada a estas funciones antes de cualquier <HTML> o <HEAD>. Esta es una restricción de las cookies no de PHP.

Todos los argumentos excepto el nombre son opcionales.

- **Nombre**. Nombre de la cookie. Cada cookie de un dominio debe tener un nombre diferente, para que el navegador pueda mantener todas las cookies separadas. Si creamos una cookie solamente con el nombre, en el cliente se eliminara la cookie que exista con ese nombre. También podemos reemplazar cualquier argumento con una cadena vacía ("").
- **Value**. Datos que almacenará la cookie en el cliente. Estos datos sólo pueden ser strings o números, y como máximo de 4KB.
- **Expire**. El argumento expire es un argumento entero que indica la fecha hora en que se eliminara la cookie en el formato de hora que devuelven las funciones UNIX time() y mktime(). Normalmente se usa time() + N. segundos de duración, para especificar la duración de una cookie.
- **Dominio**. Dominio en donde tiene valor la cookie. Si ponemos como dominio www.domain.com la cookie no se transmite para domain.com, mientras que si ponemos domain.com la cookie se transmite tanto para domain.com como para www.domain.com
- **Path**. Es la ruta del dominio a donde se envía la cookie. Si la ruta es /imagenes/, y el dominio es ejemplo.com, la cookie sólo se enviará al servidor si el navegador solicita un archivo de ejemplo.com/imagenes/. Si path es /, la cookie se enviará al servidor independientemente de la localización del archivo solicitado en el servidor.
- **Secure**. El argumento secure indica que la cookie solo se transmitirá a través de una conexión segura HTTPS.

**Ejemplo**

```
$nombre = 'usuario';
$valor = 'Homer';

// El tiempo de expiración es en 30 minutos. PHP traduce la fecha al formato
adecuado
$expiracion = time() + 60 * 30;

// Ruta y dominio
$ruta = '/cookies/';
$dominio = "prueba.dev";

// Sólo envía la cookie con conexión HTTPs
```

```
$seguridad = false;

// Cookie disponible sólo para el protocolo HTTP
$solohttp = true;

setcookie($nombre, $valor, $expiracion, $ruta, $dominio, $seguridad,
$solohttp);

echo "Cookie establecida";
```

La opción de $seguridad fuerza a que la cookie sólo sea enviada si se ha establecido una conexión segura HTTPs. $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. Esto no significa que no se puedan enviar cookies a través de HTTPs.

La cookie anterior puede ser escrita de forma mucho más reducida:

```
setcookie('usuario', 'Homer', time()+60*30, '/cookies/', 'prueba.dev',
false, true);
```

### 7.3.4 Leer contenido Cookie

Para leer el contenido de una cookie sólo tenemos que usar el array asociativo:

- $_COOKIE[‘variable’]

### 7.3.5 Eliminar Cookie

La forma más sencilla de eliminar una cookie es volviéndola a declarar pero sin la asignación de ningún valor:

```
Setcookie(‘variable’)
```

Pero este método a veces falla, por ejemplo algunos navegadores no borran la cookie sino coincide el path, por ello la forma más segura de borrar una cookie es colocar una nueva con el mismo nombre, sin valor, mismo path y nombre de dominio si se especificaron y con un tiempo de vida negativo, por ejemplo time() – 3600

```
setcookie("nombre", ‘’, time()-3600)
```

### 7.3.6 Ejemplos de uso Cookies

#### *7.3.6.1 Registro número de visitas*

```
<?php
If (isset($_COOKIE['visitante']))
{
      $numero=$_COOKIE['visitante'];
      $numero+=1;
}
```

```
else $numero=1;
setcookie("visitante",$numero,time()+86400);
print "Es la $numero ª vez que visitas esta página";
?>
```

#### 7.3.6.2 Varios valores a una misma cookie

Podemos asignar **varios valores a una misma cookie**, aunque realmente se crean tantas cookies como valores se introducen

```
setcookie("Libro[0]","El médico",time()+300);
setcookie("Libro[1]","Noah Gordon",time()+300);
setcookie("Libro[2]","1992",time()+300);
```

#### 7.3.6.3 Cookies con valores tipo array

Es posible crear cookies en las que la variable contiene un array de tipo escalar o de tipo asociativo. Es necesario incluir un setcookie por cada uno de los valores del array aunque a la hora de devolver esa variable el navegador del cliente envía el array completo

```
<?
      $valores=Array("Verde","Verano","Rolls-Royce","Millonario");
      setcookie("cookie3[color]",$valores[0],time()+3600);
      setcookie("cookie3[estacion]",$valores[1],time()+3600);
      setcookie("cookie3[coche]",$valores[2],time()+3600);
      setcookie("cookie3[finanzas]",$valores[3],time()+3600);
      if (isset($_COOKIE['cookie3']) ) {
            foreach($_COOKIE['cookie3'] as $indice=> $valor) ) {
                  echo "$indice == $valor\n";
            }
      }
?>
```

#### 7.3.6.4 Acceso inmediato a la cookie

Si deseamos acceder inmediatamente a la cookie, podemos recargar la página antes de que esta sea enviada al usuario.

```
<?
// Creamos una variable de control para verificar que la cookie se ha
creado
      If (¡isset($COOKIE_SET)){
            // Si no existe $COOKIE_SET creamos la cookie
            Setcookie('saludo', 'hola mundo');
            // Creamos la variable COOKIE_SET y le asignamos 1
            Header('Location: $PHP_SELF?COOKIE_SET=1');
            Exit;
      ]
?>
```

#### 7.3.6.5 Comprobar si las cookies están activadas

```
<?php
      setcookie("test_cookie", "test", time() + 3600, '/');
?>
```

```
<html>
<body>

<?php
      if(count($_COOKIE) > 0) {
            echo "Cookies are enabled.";
      } else {
            echo "Cookies are disabled.";
}
?>
```

### 7.3.7 Consideraciones Específicas de las Cookies

El uso de las cookies puede presentar limitaciones. De manera general, se considera que una cookie no debería superar los 4 Kb. Sin embargo, una cookie de más de 2Kb puede acarrear inconvenientes también. Además de la limitación en tamaño, hay que contar con la cantidad de cookies posibles en el disco duro del cliente el cual no debería superar los 20 archivos. Esto se debe a que el propio sistema del cliente no puede memorizar tantos enlaces a la vez. Se trata de una limitación implementada por razones de seguridad funcional. Si se utiliza algún tipo de cifrado para un valor y lo almacena dentro de la cookie, también hay que tener mucho cuidado. La cantidad de caracteres que genera un cifrador (algoritmo de encriptación o similar) puede ser muy grande y superar la cuota de tamaño dentro de la cookie, lo cual inevitablemente, truncará la información. Por lo tanto, es recomendable evitar inflar demasiado las cookies.

Otro problema es el código o juego de caracteres. El uso de un juego apropiado para un determinado país puede ser perjudicial para otro país. Es por ello que es recomendable utilizar un juego de caracteres estandarizado, por ejemplo, la norma ANSI '94. Esta norma puede sacrificar ciertos tipos de caracteres, pero seguramente no fallará en las máquinas de los clientes.

Otro gran inconveniente es el bloqueo de las cookies por parte de los navegadores. El bloqueo se suele utilizar como medida preventiva o de seguridad. Este hecho implica varios inconvenientes. Obligar al usuario a habilitar las cookies para que nuestras cookies se puedan almacenar en el disco de la computadora cliente. Al no poder registrarse la cookie en la máquina cliente, resulta entonces, imposible gestionar las sesiones. Este problema directamente puede tener muchas lecturas negativas. Muchos usuarios podrían ignorarlas por falta de conocimiento o por miedo a los tan temidos virus informáticos y otras formas de fraude informático.

## 7.4 Cabeceras en Documentos HTML

Hemos comentado anteriormente imprescindible escribir la función setcookie() antes que las etiquetas <HTML> o <HEAD>, incluso sin poner delante espacios ni líneas en blanco, para que sea la primera que se ejecute al procesarse el script PHP antes de que el servidor envíe alguna salida al cliente.

Las cabeceras contienen diversos datos, unos para ofrecer información y otros para que la página se muestre con determinadas características. Por ejemplo, pueden incluir textos que deben aparecer, claves para que los buscadores puedan encontrarlas, información sobre los

créditos de la aplicación, control de la memoria caché de navegador, especificación del tiempo de refresco de la página, etcétera.

En lenguaje HTML una cabecera define una propiedad de la página y los atributos de esa propiedad. En este lenguaje cada cabecera va precedida de la palabra META. Por ejemplo,

```
<META NAME="Autor" CONTENT="J.J. Caballero">.
```

En la dirección http://www.w3.org/Protocols/rfc2068/rfc2068 y en la dirección http://vancouver-webpages.com/META/ hay información sobre las etiquetas que se usan en HTML dentro de las cabeceras y para qué sirven.

### 7.4.1 Funcion header() en PHP

PHP dispone de la función header() para enviar cabeceras HTTP desde este lenguaje. Su sintaxis es sencilla:

```
header(cadena con el nombre de la cabecera y su valor);
```

Al igual que pasa con setcookie(), para que funcione correctamente, hay que poner esta instrucción al principio del script, antes de que se envíe ninguna salida al navegador del cliente.

Por ejemplo:

```
header("Expires: Mon, 20 Jul 2005 22:00:00 GTM");
```

Establece que el documento HTML generado con esta cabecera y presente en la caché de un servidor proxy o del mismo navegador del cliente expira el día 20 de julio de 2005 a las 10 de la noche.

Conviene observar que el nombre de la etiqueta (Expires) es propio del lenguaje HTML y, por tanto, para saber cuáles son y lo que hace cada una, se puede recurrir a la segunda dirección de Internet que hemos indicado más arriba.

Otro ejemplo:

```
header("Content-Type: text/html");
```

Establece el tipo de documento que será válido en el navegador del usuario, en este caso los documentos de tipo texto o html.

En las dos cabeceras siguientes especificamos que la página devuelta no se guarde en la memoria de la caché del ordenador del cliente, sino que se refresque cada 10 segundos, accediendo siempre a la misma dirección.

```
header("Cache-Control: no-cahe, must-revalidate");
header("Refresh: 10; URL=".$_SERVER["PHP_SELF"]);
```

### 7.4.2 Redireccionar con header()

Una de las grandes utilidades de header() es la de redireccionar el flujo de control de una aplicación PHP.

La sintaxis es la siguiente

```
header("Location: direccion");
```

Veamos el siguiente ejemplo:

```
header("Location: validar.php");
```