

Esta unidad trabaja temas relacionados con la seguridad de una página web y ciertos mecanismos que pueden ser implementados en el lado del servidor para garantizar dicha seguridad.

En el **apartado 5.1 Mantenimiento del estado en aplicaciones web** se comenta que la web se basa en un modelo no conectado (stateless, "sin estado") ya que su funcionamiento está basado en el protocolo http que no es capaz de recordar información sobre las páginas visitadas durante la navegación por una aplicación web.

Sin embargo, estamos acostumbrados a que esto no sea así; tal es el caso de una compra online donde el carro de la compra se mantiene actualizado, la necesidad de no tener que registrarnos en muchas aplicaciones para acceder a ellas (es suficiente con hacerlo la primera vez, poder seguir con un juego sin necesidad de comenzar de nuevo en él,...)

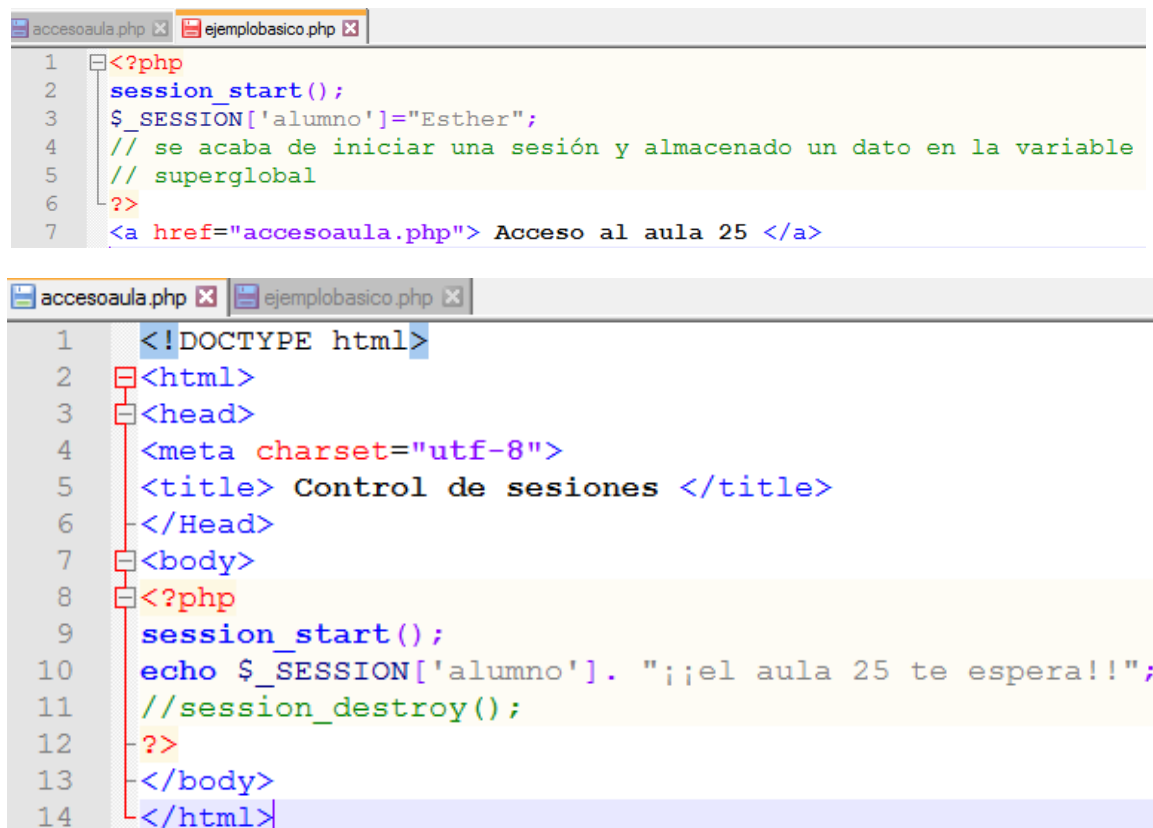
El control del estado de una aplicación web, es gracias a la implementación en las aplicaciones web de mecanismos que permiten un control en el acceso a ella por parte de un usuario; es lo que se denominada, un control de sesión. Con estos mecanismos se pretende almacenar información de cada usuario mientras navega por una aplicación web. En la sesión se establece una comunicación entre el cliente web y el servidor web, pudiéndose establecer un control sobre ella o bien a través de la URL (los datos de la sesión se pasan a través de ella) o bien a través de la creación de cookies (los datos de la sesión se incluyen en un archivo)(en son excluyentes).

Cuando un cliente web realiza una petición sobre un servidor web solicitando una aplicación web, el servidor responde con un código único de identificación que es la clave para el mantenimiento de las sesiones; se trata del SID.

A continuación, veremos cómo controlar una sesión a través de php (entorno servidor)

1. Control de sesión a través de la URL.

Ejemplo sencillo para entender las funciones:



```

1  <?php
2  session_start();
3  $_SESSION['alumno']="Esther";
4  // se acaba de iniciar una sesión y almacenado un dato en la variable
5  // superglobal
6  ?>
7  <a href="accesoaula.php"> Acceso al aula 25 </a>

accesoaula.php x ejemplobasico.php x

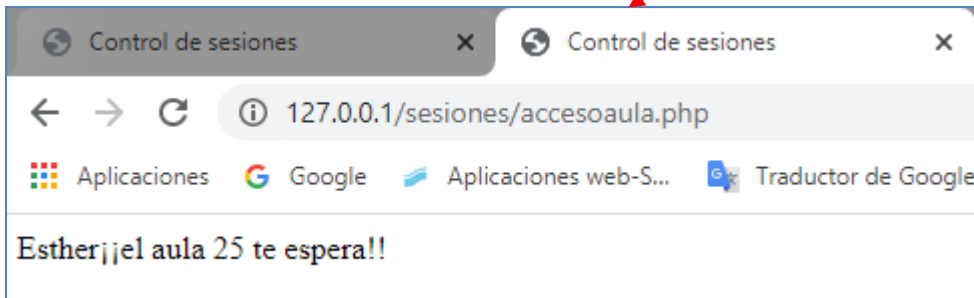
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <meta charset="utf-8">
5  <title> Control de sesiones </title>
6  </Head>
7  <body>
8  <?php
9  session_start();
10 echo $_SESSION['alumno']. "¡¡el aula 25 te espera!!";
11 //session_destroy();
12 ?>
13 </body>
14 </html>

```

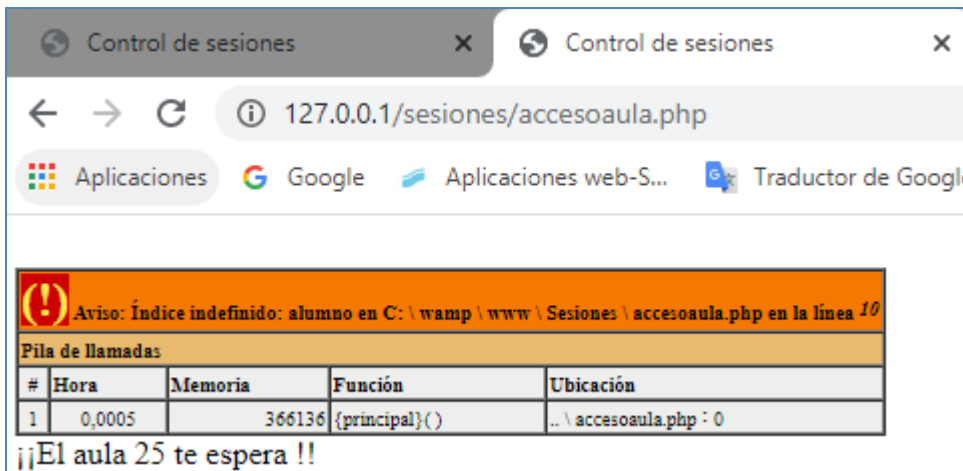
Al ejecutar la aplicación web `ejemplobasico.php`, se crea un SID y al navegar por las páginas de la aplicación la función `session_start()` recupera la sesión y con ello los datos almacenados en la variable `$_SESSION`.

Sin embargo, la sesión ha quedado abierta y es posible acceder a `accesoaula.php` de forma directa (vulnerabilidad no controlada).

Ejecución de la aplicación en
dos pestañas del navegador

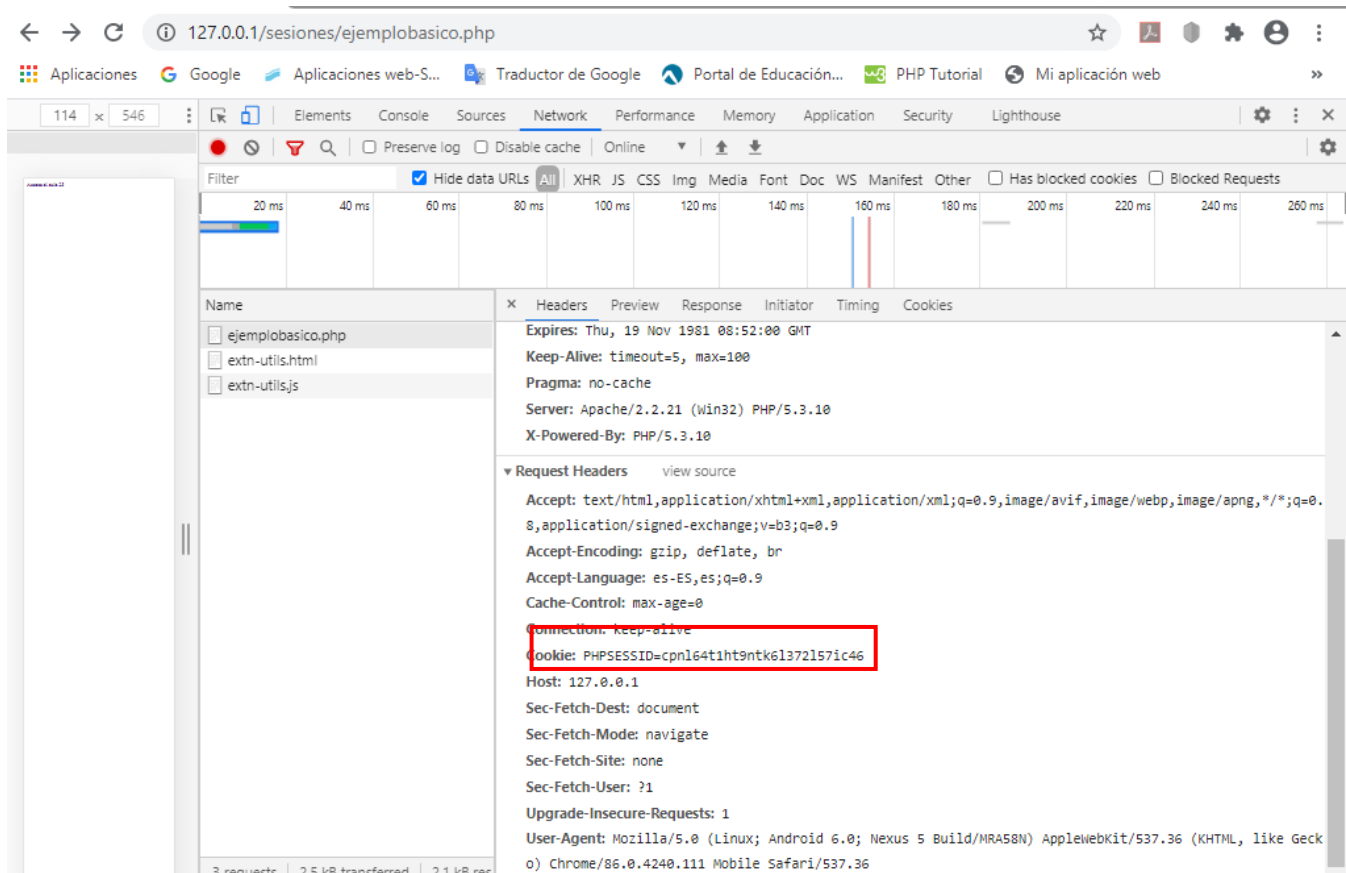


Para ello, se debe cerrar la sesión con la función `session_destroy()`, con lo cual el acceso ahora a `accesoaula.php` dará un error ya que en el primer acceso se destruyó la sesión y en segundo acceso se quiere acceder a una variable (`$_SESSION['alumno']`) que no existe.



Sin embargo, si se cierra la aplicación web, la sesión también se cierra automáticamente (ello es debido a la configuración de ciertas variables del fichero de configuración de `php.ini`).

Y ¿dónde se almacena el SID de la sesión? En una cookie “especial” (cookie de sesión)



Lo cual se puede ver inspeccionando las **cabeceras del protocolo HTTP** a través de los navegadores web por ejemplo. (en Chrome-Más herramientas-herramientas para desarrolladores-Application-Session Storage-Network-Headers)

2. mensajes del protocolo HTTP



HTTP es un protocolo sin estado que utiliza TCP como protocolo de transporte y determina los tipos de peticiones que los clientes pueden enviar, así como el formato y estructura de las respuestas. Define también una estructura de metadatos, en forma de cabeceras que se envían tanto en las peticiones como en las respuestas.

Una vez que el navegador analiza la URL y establece la conexión TCP con el servidor web, el navegador envía un **mensaje HTTP de petición** que depende de la URL. El servidor enviará un **mensaje de respuesta** que depende de la petición enviada y del estado del servidor. Después se cerrará la conexión TCP.

Los mensajes que utiliza HTTP se componen de líneas escritas en texto plano (ASCII) que contienen las órdenes y los parámetros con la sintaxis definida por el protocolo. Pueden ser de dos tipos: **mensajes de petición** y **mensajes de respuesta**.

Mensajes de petición. Hay siete diferentes. Su estructura es la siguiente:

- Línea inicial de petición:

Método utilizado (GET, POST,...)

El servidor proxy si se utiliza para la conexión a internet

La versión del protocolo utilizada (HTTP 1.1, HTTP 2.0,...)

- Líneas de cabecera: Conjunto de pares nombre/valor denominados cabeceras (cada cabecera se muestra en una línea; si no hay cabeceras 0). Se pueden clasificar en cuatro grupos (generales, de petición, de respuesta, de entidad)
- Cuerpo del mensaje (opcional): Contiene parámetros o ficheros a enviar al servidor.

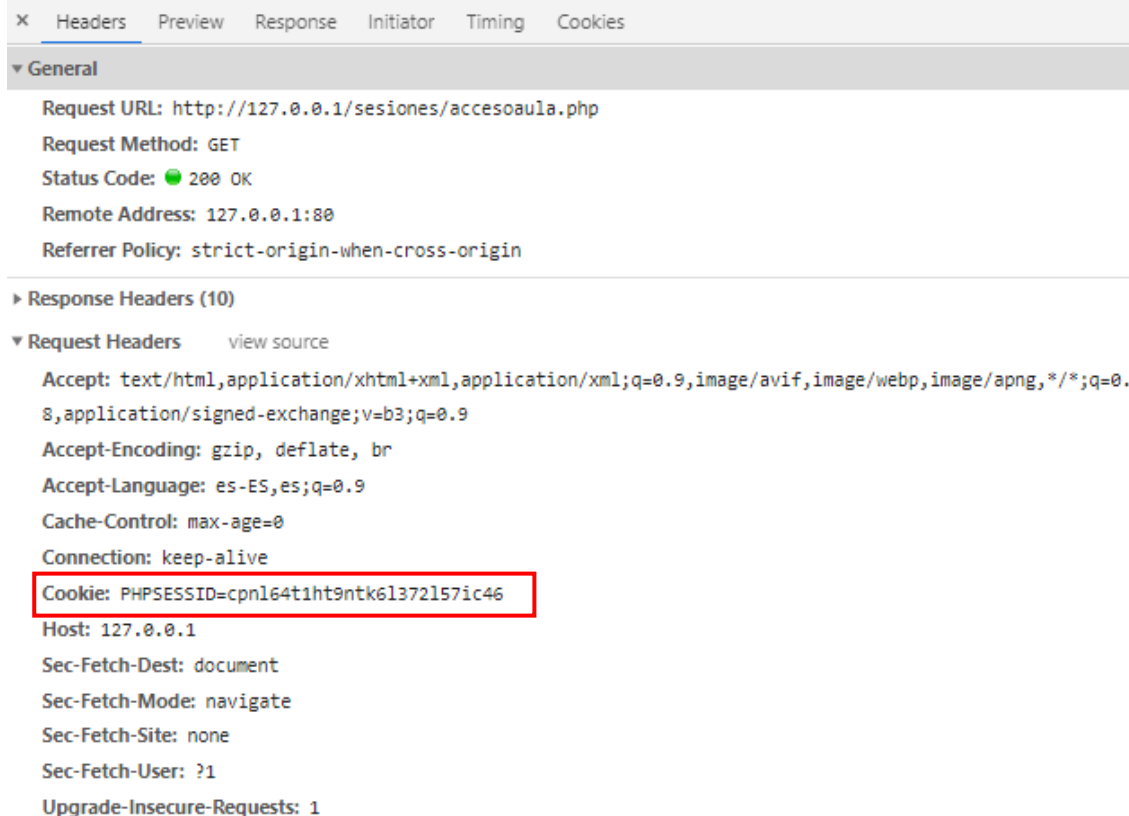
Mensaje de respuesta. En ellos se envían códigos de estado y error por el servidor que informan al cliente de cómo ha sido procesada la petición. También se informa del uso de caché, redirección a servidores proxy, compresión de los recursos antes de enviarlos a los clientes, si se envían cookies por el servidor para poder ser utilizada por el cliente (navegador) en solicitudes posteriores al servidor, etc.

La **cookie es un archivo que almacena información sobre la sesión** (cookie de sesión; son almacenadas en el cliente web a petición del servidor web. Existen mientras la sesión esté vigente. Al cerrar el navegador, se destruyen.)

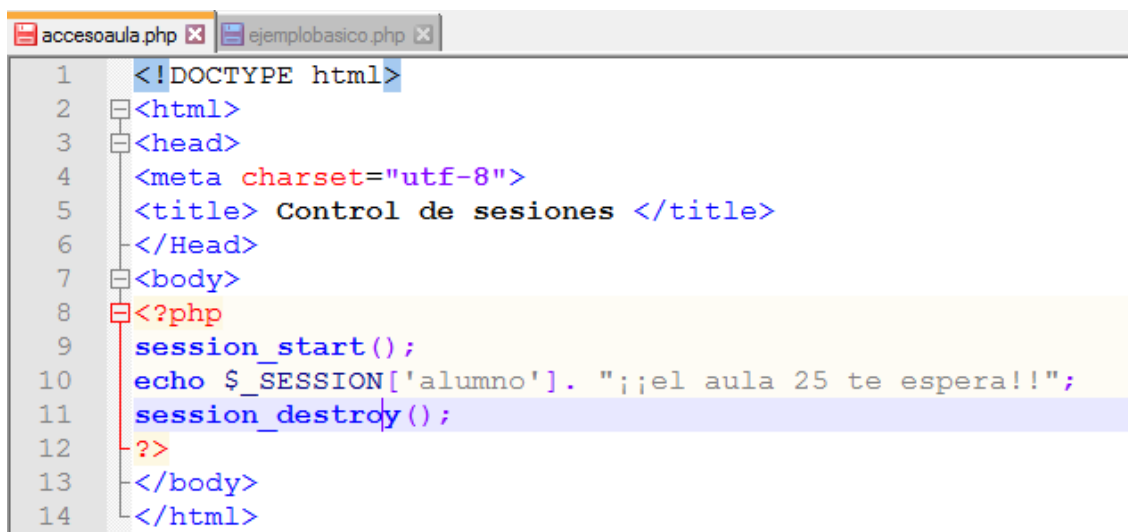
Para ver detalles de está cookie, acceder a Network-cookies.

Request Cookies <input type="checkbox"/> show filtered out request cookies									
Name	Value	Domain	Path	Expires / Max-Age	Si...	Http...	Secure	Sa...	Priority
PHPSESSID	cpnl64t1ht9ntk6l372l...	127.0.0.1	/	Session	35				Medium

Si accedemos a accesoaula.php y analizamos la información de los encabezados de http ahora veremos que el SID es el mismo (session_start mantiene la sesión)



Si se hubiera cerrado la sesión, se hubiese destruido la información almacenada también en la cookie.



Nota: en la variable `$_SESSION`, se almacenan las variables de la sesión (antiguamente se realizaba a través de la función `session_register()` que está obsoleta. La diferencia está en que con esa función, no había que incluir `session_start` previamente.) Con `$_SESSION` si se quiere tampoco hay porqué hacerlo si se tiene la variable `session.auto start=1` del fichero `php.ini` (no es conveniente; por defecto está a 0).

La función `session_unregister()` tb está obsoleta. Utilizar `unset($_SESSION['variable'])` y se eliminará esa variable de la sesión. Si se quieren eliminar todas las variable, se puede utilizar `session_unset` lo cual no es lo mismo que `session_destroy()` ya que anteriormente se pierden las variables de la sesión pero la sesión sigue vigente (el SID se mantiene y con `session_destroy` no)

3. COOKIES.

Hemos visto un tipo de COOKIES (cookie de sesión). Otro tipo de COOKIES son las persistentes, es decir existen durante más tiempo.

Una cookie persistente es un archivo que el servidor web envía al equipo de quien accede a un sitio web.

Las cookies son necesarias para ayudar al usuario en la navegación dentro de una aplicación web y evitar que tenga que logearse en más de un punto de la navegación y para almacenar información sobre sus accesos anteriores con el fin de recuperar datos necesarios para reanudar un proceso iniciado en visitas anteriores (ej: carro de la compra de una tienda online). Los fines deben ser estos aunque en la realidad hoy un fin es publicitario. De ahí que haya sido necesario dictar una ley que regule su uso para evitar un uso fraudulento. (<https://ayudaleyprotecciondatos.es/2018/03/05/guia-uso-cookies/>)

En una cookie se puede almacenar mucha información y ser rescatada por la aplicación web cuando se quiera. Si en una cookie se almacenan los hábitos de navegación de un usuario, es decir, las url que más frecuenta el usuario, ello es utilizado por las empresas publicitarias (cookies a terceros) para después mostrarnos banner publicitarios. Por ejemplo, si accedes a una página de una determinada temática, esa página, crea una cookie para ti. Si al día siguiente, vuelves a acceder a esa misma página, se recupera la cookie creada y es habitual que en ese segundo acceso salgan pantallas emergentes con publicidad sobre contenidos sobre esa temática. Por eso motivo, en Europa se dicta un reglamento (UE 2016/679) que establece las bases para la redacción de las Leyes de Protección de Datos Personales y garantía de los derechos digitales (en España, ley orgánica del 3/2018, 5 de diciembre) que regula el uso de las cookies. El 31 de octubre de 2020, todas las páginas web de acceso público, deben cumplir con esta normativa).

Una cookie puede existir en el disco duro del equipo del usuario desde el cual se accede a una determinada aplicación web (cookie persistente) o Sólo durante la sesión o acceso a la aplicación web (cookie de sesión) y se almacena en la RAM.

Al crear una cookie en el servidor se establecen una serie de parámetros:

Duración:

La vigencia de la cookie se establece con el parámetro de duración de la cookie. Por defecto su vigencia es durante el tiempo que esté ejecutándose el archivo que la contenga (es decir, si cierro la pestaña del navegador desde la cual se invocó al recurso web, fichero .php, que la pone en ejecución, dejará de estar vigente)

Su vigencia puede ser modificada utilizando como parámetro que establece la duración de la cookie, la función time()+X segundos. (si en vez de + X segundos es -X segundos, la cookie durará incluso menos)

Directorio de actuación:

Por defecto el directorio en el cual está incluido el archivo que contiene la creación de la cookie y los subdirectorios de éste. Ahí es donde actuará.

Dominio:

Al igual que antes, se puede establecer, en un 5 parámetro, en qué dominios o subdominios tiene vigencia una cookie (en nuestro caso como trabajamos en local, no tiene mucho sentido este parámetro pero si nuestra aplicación pertenece a un determinado dominio sí.

Secure:

Con este parámetro se establece si la cookie debe crearse utilizando este parámetro o no (http o https)

Httponly:

Para indicar si la cookie es accesible sólo desde http o no.

CREAR UNA COOKIE: utilizando setcookie()

ELIMINAR UNA COOKIE:

Desde el propio navegador (localizar la opción de borrado de cookies; desaparecen de la caché)

Desde código php en el entorno servidor asignado tiempo negativo.

Ejemplo: crear una pequeña cookie y leerla desde otro archivo. Acceder a las cabeceras de HTTP para verla.