



# TEMA 3.6

Sesiones

## Gestión del estado: Sesiones

- Como hemos visto en el tema de cookies, las cookies facilitan la gestión del estado.
- Sin embargo, la información de las cookies se guarda en el navegador, y puede manipularse. Por tanto, **las cookies no son seguras**.
- Otra opción para gestionar el estado son las sesiones.
- Una sesión es el intercambio de información entre la aplicación en servidor y un navegador cliente. Cada vez que un cliente concreto haga un request, la aplicación sabrá quién es gracias a la sesión.
- Las **sesiones** se almacenan en el **servidor**, lo que dificulta una manipulación externa.
- Cada sesión tiene asociado un almacén de datos (en servidor), cuya información se manipula mediante la superglobal **`$_SESSION`**.

## Cookies de sesión: Identificador único

- En la mayoría de las aplicaciones del lado de servidor, cuando se recibe un request sin identificación, se crea una **sesión**.
- Al crear la sesión, PHP genera *automáticamente* una **cookie de sesión**, que será enviada al navegador en el response.
- En PHP la cookie de sesión se llama **PHPSESSID** y su valor es un identificador. Este número identificará de forma única la interacción entre navegador y servidor. Así el servidor podrá distinguir entre todas las sesiones que tiene activas en ese momento.
- La información relacionada con la sesión se almacena asociada a ese identificativo único. La superglobal **\$\_SESSION** nos permite añadir y consultar la información asociada a esa sesión.
- La información que se guarda en una sesión puede ser tanto el *login* del usuario, como un carrito de la compra.

## Ejercicio 1:

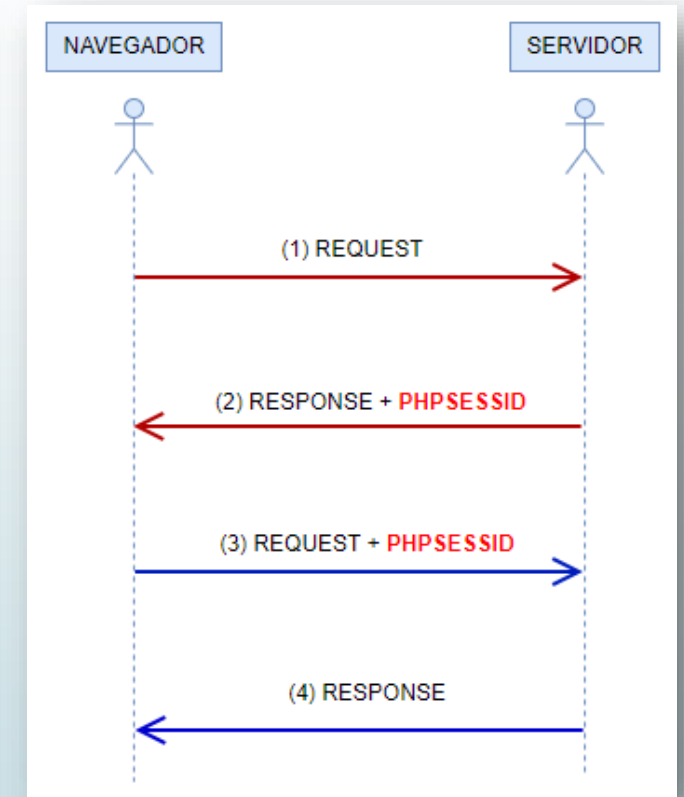
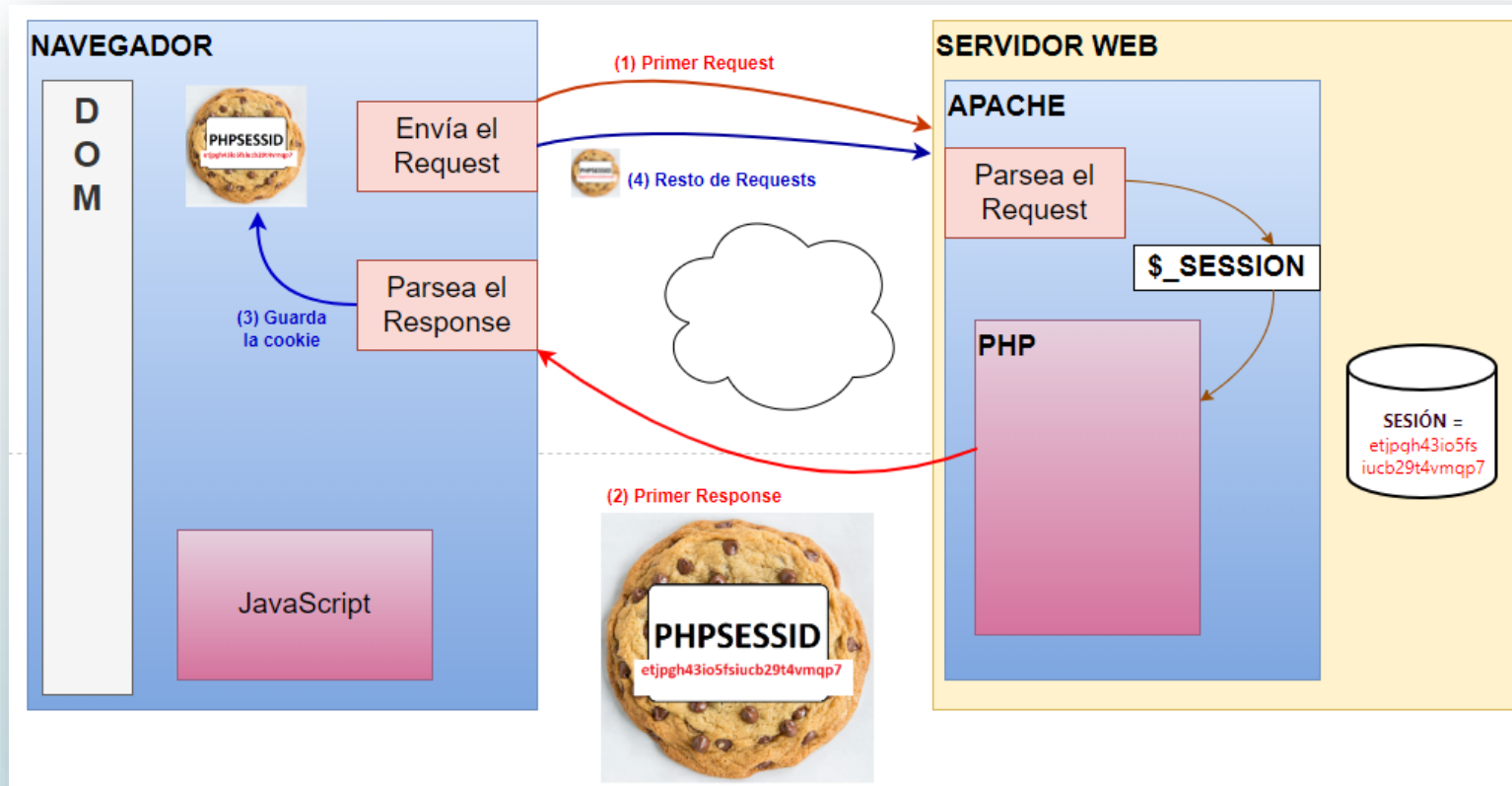
# Cookie de sesión

```
set-cookie: PHPSESSID=etjpg43io5fsiucb29t4vmqp7; path=/
```

PHPSESSID	etjpg43io5fsiucb29t4vmqp7	www.tsugi.org	/	Session
-----------	---------------------------	---------------	---	---------

1. Abre una nueva pestaña en tu navegador, y activa la pestaña de red del inspector.
2. Conéctate a la web:  
<https://www.tsugi.org>
3. Observa los headers del response, si es la primera vez que te conectas al sitio, aparecerá un **set\_cookie**
4. El nombre de la cookie es **PHPSESSID**, pues es una cookie de sesión. Su valor es el identificador de tu sesión con tsugi.
5. Activa la pestaña aplicación del inspector, y verás la cookie PHPSESSID asociada a tsugi.org

# Cookies de sesión: Resumen

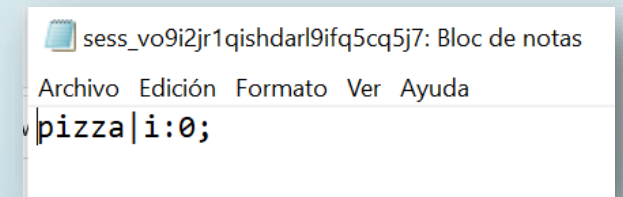
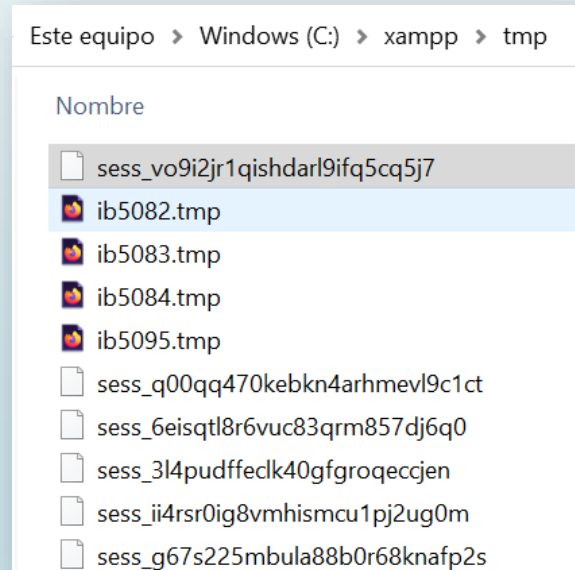


- (1)** Cuando llega un primer request, la aplicación comprueba si está identificada la sesión, y si no genera una con un **identificador único**. La sesión se almacena en el servidor (en un directorio temporal). PHP la gestionará en la superglobal **\$\_SESSION**.
- (2)** En el response la aplicación PHP envía al navegador la cookie de sesión, su nombre es PHPSESSID, y su valor el identificador único.
- (3)** El navegador almacena la cookie de sesión (**PHPSESSID=identificador único**).
- (4)** Siempre que el navegador haga un request a ese servidor, enviará sus cookies, incluida la de sesión (PHPSESSID). Al parsear la request, el servidor reconocerá la sesión y cargará los datos en **\$\_SESSION**.
- (5)** La sesión sólo podrá crearse o eliminarse en el servidor. Los datos de sesión nunca viajan al navegador, se guardan siempre en el servidor.

# Configuración de sesiones en el servidor

- En localhost -> PHPInfo tenemos la configuración de las sesiones. Ahí se definen los parámetros para el manejo de sesiones por PHP
- Un parámetro es el directorio temporal donde se almacenan los ficheros de sesión (save\_path)
- Los ficheros de sesión no se eliminan salvo que lo pidamos expresamente desde nuestra aplicación. Almacenan pares clave=>valor, los correspondientes a \$\_SESSION
- El nombre de la cookie de sesión es por defecto PHPSESSID, pero podríamos modificarlo.

session.name	PHPSESSID
session.referer_check	no value
session.save_handler	files
session.save_path	C:\xampp\tmp
session.serialize_handler	php
session.sid_bits_per_character	5
session.sid_length	26
session.upload_progress.cleanup	On
session.upload_progress.enabled	On



# Código fuente

Las funciones para manipular sesiones con PHP son:

- **session\_start**: Se llama al principio del fichero PHP. Si existe una sesión, los datos se cargarán en `$_SESSION`. Si no existe la sesión, se creará una, así como la cookie PHPSESSID.
- **session\_destroy**: Elimina el fichero físico de la sesión, pero no la cookie de sesión. Para eliminar la cookie tendremos que utilizar `setcookie` (Indicando el nombre de la cookie, que es PHPSESSID, y una fecha de caducidad en el pasado, por ejemplo `time()-3600`)

```
<?php
session_start();
if ( ! isset($_SESSION['pizza']) ) {
    $mensaje = "La Sesión está vacía";
    $_SESSION['pizza'] = 0;
} else if ( $_SESSION['pizza'] < 3 ) {
    $_SESSION['pizza'] = $_SESSION['pizza'] + 1;
    $mensaje = "Sumamos 1 ...";
} else {
    session_destroy(); // Borra el fichero
    session_start(); // Crea un nuevo fichero. Estará vacío.
    $mensaje = "Session reiniciada con destroy-start";
}
?>
```

1. Esta aplicación almacena un valor "pizza":
  1. La primera vez que se pide esta página, se inicializa "pizza" a 0
  2. En sucesivos request (de la misma sesión), se incrementa "pizza" en 1.
  3. Si "pizza" es mayor o igual que 3, se destruye la sesión, iniciando una nueva.
2. Con **session\_start** el programa comprueba si hay una cookie de sesión.
  1. Si la hay restaura la información de la sesión en **\$\_SESSION**. Si no la hay crea una nueva sesión y una nueva cookie de sesión. En **\$\_SESSION** estará la información de "pizza", y la cookie se llamará **PHPSESSID**.
  2. Si había una sesión, PHP cargará en **\$\_SESSION** los datos de la misma. En mi aplicación, el dato que se almacena es "pizza".
3. Con **session\_destroy** se destruye la sesión (el fichero) y se inicia otra (con otro código), modificando el valor de la cookie PHPSESSID.



## Manipulación de sesiones:

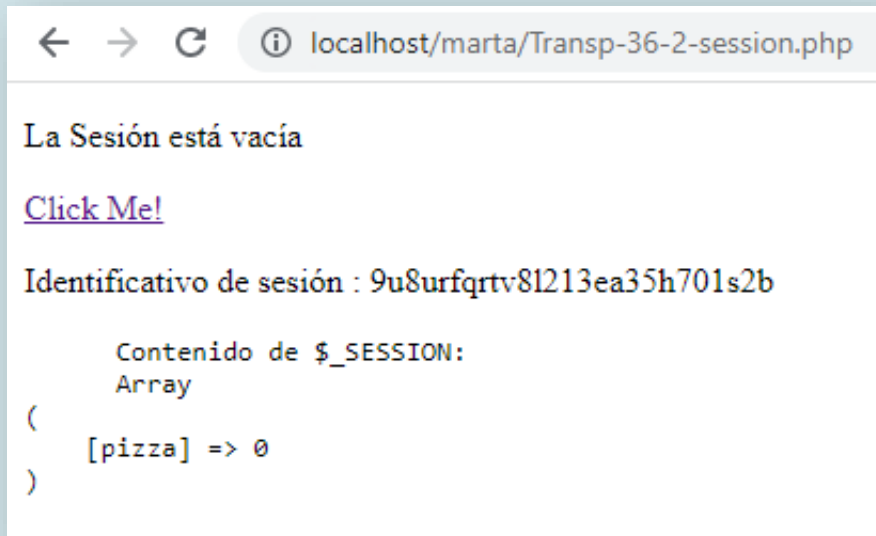
Para manipular sesiones en PHP utilizaremos:

- **session\_start**: Carga la sesión si existe. Si no existe, la crea
- **session\_destroy**: Destruye la sesión
- **session\_id**: Retorna el código de la sesión
- **\$\_SESSION["clave"] = "algo"**: Añade la información "clave" con valor "algo" al fichero de la sesión. Si ya existía modifica el valor con el nuevo.
- **unset(\$\_SESSION["clave"])**: Elimina la información correspondiente a "clave". Desaparecerá del fichero de sesión.



## Ejercicio 2:

## Sesión



A screenshot of a web browser window. The address bar shows 'localhost/marta/Transp-36-2-session.php'. The page content includes the text 'La Sesión está vacía', a blue underlined link 'Click Me!', and a session identifier 'Identificativo de sesión : 9u8urfqrtv81213ea35h701s2b'. Below this, it displays the contents of the \$\_SESSION array, which currently contains a single entry: '[pizza] => 0'.

```
← → ↻ ⓘ localhost/marta/Transp-36-2-session.php

La Sesión está vacía

Click Me!

Identificativo de sesión : 9u8urfqrtv81213ea35h701s2b

Contenido de $_SESSION:
Array
(
    [pizza] => 0
)
```

- Implementa el código ejemplo de estas transparencias. El valor de pizza debe incrementarse hasta 3 y luego resetearse a 0.
- Pruéba el código recargando la página mediante el enlace "Click Me!"
- Investiga:
  1. ¿Por qué no crea un nuevo identificador de sesión cuando haces `destroy_session`?
  2. ¿Puedes, sin modificar el programa, conseguir que cambie el identificador de sesión?
  3. ¿Qué pasa con los ficheros temporales de sesión si haces la trampa del punto anterior?
- Modifica el programa para añadir información a la sesión: Incluye una entrada "bebida" con valor de tipo string, y una entrada "postre", que será un array asociativo, la clave el nombre de un postre y el valor su precio.

## Sesiones sin cookie de sesión:

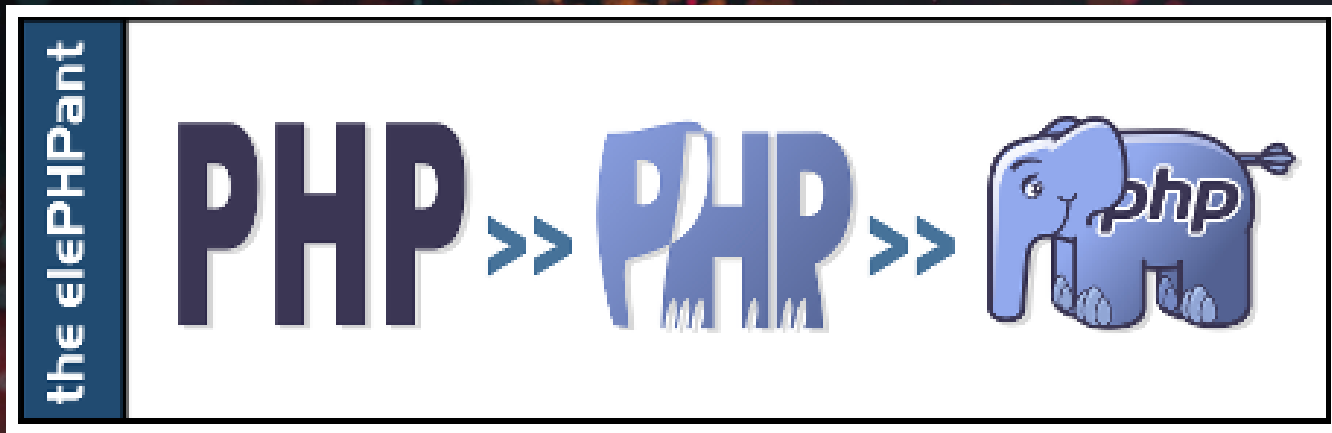
- Muchas aplicaciones utilizan sesiones sin cookies de sesión.
- Esto es así porque, mediante la cookie de sesión, no podremos tener instancias (sesiones) distintas en pestañas diferentes del navegador.
- Para un carrito de la compra, o para logines, podemos utilizar cookies de sesión. Esto mantendrá la misma sesión en pestañas distintas del navegador.
- Si lo que necesitamos es que la aplicación funcione en un iframe, o tener distintas instancias de la aplicación en distintas pestañas, las cookies de sesión no nos sirven.
- PHP permite gestionar sesiones sin cookies de sesión. Para ello, será necesario establecer una configuración antes del `session_start`.
- Al no existir cookie, no se mantiene el estado. En este caso puede simularse almacenando el ID de la sesión en un input oculto con persistencia, y enviarlo en un `$_GET` o un `$_POST`. No es seguro, habrá que resolverlo.

```
<?php
// Configuramos PHP para no utilizar cookies de sesión
ini_set("session.use_cookies","0");
ini_set("session.use_only_cookies", "0");
ini_set("session.use_trans_sid","1");
session_start();
IF (!isset($_SESSION["holamundo"]))
    $_SESSION["holamundo"] = "HOLA!!";
?>
<pre>
    <?=session_id()?>
    <?php print_r($_SESSION) ?>
</pre>
```

- En este ejemplo, no tenemos una cookie de sesión (PHPSESSID), por lo que no hay forma establecer un estado. El ID de sesión siempre será distinto.
- Si queremos mantener el estado, tendremos que hacer persistencia en un campo que guarde el id de sesión, con un `$_POST` o un `$_GET`.

← → ↻ ⓘ localhost/marta/Transp-36-2-nocooksess.php

```
k3pmhgv1qv3p7gp52aoi19vv1n  Array
(
    [holamundo] => HOLA!!
)
```



<https://www.php.net/docs.php>