

Sesiones

En esta lección se trata el manejo de sesiones en PHP.

Introducción

Una de las limitaciones de las páginas web es que cada página web es un documento independiente. Eso hace que dos programas PHP no puedan, en principio, compartir información.

En principio, la única manera de enviar información de una página a otra es a través de un formulario y se trata de la información que ha introducido el usuario en ese formulario. Como mucho podemos enviar información adicional mediante controles ocultos (como se comenta en el [apartado sobre controles ocultos de la lección Varios](#)), o podemos "imitar" a un formulario redirigiendo a otra página enviando información en la dirección (como se comenta en la [lección sobre cabeceras](#)), pero esto no puede considerarse compartir información, puesto que la página que recibe la información no puede saber si la información ha sido manipulada por el camino ni quién se la envía.

Por compartir información nos estamos refiriendo a utilizar variables comunes en programas distintos, sin que la información salga del servidor.

El ejemplo siguiente muestra dos páginas: en la primera se crea una variable, que se puede utilizar en esa misma página. La segunda página, que se encuentra en el mismo servidor, no tiene acceso a la variable creada en el primer programa y por eso se produce un aviso al ejecutarla.

```
<?php
$nombre = "Pepito Conejo";
print "<p>El nombre es $nombre</p>";
```

El nombre es Pepito Conejo

```
<?php
print "<p>El nombre es $nombre</p>";
```

Notice: Undefined variable: nombre in **ejemplo.php** on line **2**

El nombre es

PHP puede superar esta limitación mediante las sesiones. Las sesiones permiten que páginas distintas puedan acceder a una variable común, la matriz \$_SESSION.

El ejemplo siguiente muestra dos páginas: en la primera se crea la variable en la matriz \$_SESSION, que se puede utilizar en esa misma página. La segunda página, que se encuentra en el mismo servidor, tiene acceso a la variable creada en el primer programa (si se ha ejecutado antes el primer programa, obviamente).

```
<?php
session_start();

$_SESSION["nombre"] = "Pepito Conejo";
print "<p>El nombre es $_SESSION[nombre]</p>";
```

El nombre es Pepito Conejo

```
<?php
session_start();

print "<p>El nombre es $_SESSION[nombre]</p>";
```

El nombre es Pepito Conejo

Las sesiones no deben confundirse con [las cookies](#). Las cookies es un método que permite guardar información en el ordenador del cliente para recuperarla en el futuro, mientras que en las sesiones la información se mantiene en el servidor hasta que se cierra la sesión (por intervención del usuario o por tiempo). En el manual de PHP se ofrece un [capítulo dedicado a las sesiones](#).

El trabajo con sesiones tiene tres partes:

- Creación o apertura de la sesión

Cuando alguna página crea una sesión utilizando la función correspondiente, el servidor asocia al navegador del usuario un identificador de usuario único. El identificador se guarda en el usuario en forma de cookie o, si el navegador del usuario no permite la creación de cookies, añadiendo el identificador en la dirección de la página.

- Utilización de la sesión

Si ya se ha creado la sesión, las páginas solicitadas por el mismo navegador pueden guardar y recuperar información en el servidor, información que se asocia al identificador de usuario, por lo que no es accesible a otros usuarios. La información se conserva hasta que el usuario o el servidor destruyan la sesión.

- Destrucción o cierre de la sesión

Tanto el usuario como el servidor pueden cerrar la sesión. El usuario puede destruir la sesión cerrando el navegador. El servidor puede destruir la sesión cuando alguna página utilice la función correspondiente o al cabo de un tiempo determinado (definido mediante la función [session_set_cookie_params\(\)](#)).

Crear la sesión

En PHP, las sesiones se crean mediante la función [session_start\(\)](#). Si la sesión no existía, esta función crea la sesión y le asocia un identificador de sesión único. Si la sesión ya existía, esta función permite que la página tenga acceso a la información vinculada a la sesión. Es decir, todas las páginas que quieran guardar datos en \$_SESSION o leer datos de \$_SESSION deben comenzar con la función **session_start()**.

```
<?php
session_start();

$_SESSION["nombre"] = "Pepito Conejo";
print "<p>El nombre es $_SESSION[nombre]</p>";
```

El nombre es Pepito Conejo

```
<?php
session_start();

print "<p>El nombre es $_SESSION[nombre]</p>";
```

El nombre es Pepito Conejo

La creación de sesiones requiere el envío de cabeceras HTTP, por lo que la función session_start() debe utilizarse antes de empezar a escribir el contenido de la página. En caso contrario PHP producirá un aviso y no se creará la sesión. El motivo es que el identificador de sesión se utiliza en las cabeceras de respuesta HTTP y las cabeceras se envían antes del texto de la página. En la [lección sobre cabeceras](#) se comenta con más detalle el problema del envío de cabeceras.

El ejemplo siguiente es incorrecto, ya que utiliza la función session_start() después de haber escrito texto.



```
<?php
// Este código es incorrecto, la sesión se crea
después de crear texto
print "<p>Hola</p>\n";
session_start();
```

Hola

Warning: session_start(): Cannot send session cache limiter - headers already sent (output started at ejemplo.php:3) in **ejemplo.php** on line 4

Nota: Como se comenta también en la [lección sobre cabeceras](#), dependiendo de la configuración de la directiva [output_buffering](#), la sesión podría crearse correctamente y no generarse el aviso.

Utilizar la sesión

Cuando una página ha creado una sesión o ha accedido a una sesión ya existente mediante **session_start()**, la página tiene acceso a la matriz **\$_SESSION** que contiene las variables de esa sesión.

La matriz **\$_SESSION** es una matriz asociativa en la que se pueden definir valores como en cualquier otra matriz. La diferencia es que **\$_SESSION** es accesible desde páginas diferentes (siempre que esas páginas tengan asociada la misma sesión), manteniéndose los valores de una página a otra.

El ejemplo siguiente muestra dos páginas. La primera página guarda información en **\$_SESSION** y la segunda la utiliza.

```
<?php
session_start();

$_SESSION["nombre"] = "Pepito Conejo";
print "<p>Se ha guardado su nombre.</p>\n";
```

<p>Se ha guardado su nombre.</p>

```
<?php
session_start();

print "<p>Su nombre es $_SESSION[nombre].</p>\n";
```

<p>Su nombre es Pepito Conejo.</p>

Los nombres de los primeros índices de la matriz **\$_SESSION** tienen que cumplir las mismas reglas que los nombres de las variables, es decir, que el primer carácter debe ser una letra o un guion bajo (_). En particular, no deben ser números ni contener caracteres no alfanuméricos.



```
<?php
session_start();

// Este código es incorrecto, ya que PHP no puede
crear un índice numérico en $_SESSION
$_SESSION[] = "Pepito Conejo";
print "<p>Se ha guardado su nombre.</p>\n";
print "<pre>"; print_r($_SESSION); print "</pre>\n";
```

Se ha guardado su nombre.

Notice: Unknown: Skipping numeric key 0 in **Unknown** on line 0
Array
(
)



```
<?php
session_start();
```

Se ha guardado su nombre.

```
// Este código es correcto, ya que el primer índice
no es numérico
$_SESSION["nombres"][] = "Pepito Conejo";
print "<p>Se ha guardado su nombre.</p>\n";
print "<pre>"; print_r($_SESSION); print "</pre>\n";
```

```
Array
(
    [nombres] => Array
        (
            [0] => Pepito Conejo
        )
)
```

Cerrar la sesión

Cerrar una sesión es destruir la matriz `$_SESSION` y el identificador de la sesión.

Las sesiones se pueden cerrar de varias maneras:

- El usuario puede cerrar la sesión simplemente cerrando el navegador (no basta con cerrar las pestañas).
- Un programa puede cerrar la sesión mediante la función [`session_destroy\(\)`](#).
- En general, las cookies tienen una duración establecida en la directiva [`session.cookie_lifetime`](#) (y el servidor puede borrar la información cuando ha pasado el tiempo indicado en segundos en la directiva [`session.gc_maxlifetime`](#)), pero la duración de una sesión en particular puede establecerse en el momento de su creación mediante la función [`session_set_cookie_params\(\)`](#) (tiempo que se puede modificar posteriormente).

Cuando se destruye una sesión, el programa que ha destruido la sesión sigue teniendo acceso a los valores de `$_SESSION` creados antes de la destrucción de la sesión, pero las páginas siguientes no. Si se ejecuta el primero de los ejemplos siguientes y después el segundo, se obtienen los resultados indicados:

```
<?php
session_start();

$_SESSION["nombre"] = "Pepito Conejo";
session_destroy();

if (isset($_SESSION["nombre"])) {
    print "<p>Su nombre es $_SESSION[nombre].</p>\n";
} else {
```

```
<p>Su nombre es Pepito Conejo.</p>
```

```
print "<p>No sé su nombre.</p>\n";
}
```

```
<?php
session_start();

if (isset($_SESSION["nombre"])) {
    print "<p>Su nombre es $_SESSION[nombre].</p>\n";
} else {
    print "<p>No sé su nombre.</p>\n";
}
```

<p>No sé su nombre.</p>

Nombre de sesión

En principio, cuando el navegador se conecta a un servidor, la sesión es única, es decir, todas las páginas del mismo dominio compartirán la misma matriz `$_SESSION`. La función [session_name\(\)](#) permite establecer un nombre de sesión específico, de manera que todas las páginas que declaren el mismo nombre de sesión accederán a la misma matriz `$_SESSION` y las que tengan nombres de sesión distintos accederán a matrices `$_SESSION` diferentes.

En el ejemplo siguiente, los dos primeros programas crean la misma variable en `$_SESSION`, pero como se ha utilizado la función **session_name()** con nombres diferentes, realmente lo hacen en matrices `$_SESSION` distintas.

```
<?php
session_name("ejemplo1");
session_start();

$_SESSION["nombre"] = "Pepito Conejo";
print "<p>El nombre es $_SESSION[nombre]</p>";
```

El nombre es Pepito Conejo

```
<?php
session_name("ejemplo2");
session_start();
```

El nombre es Juan Fuláñez

```
$_SESSION["nombre"] = "Juan Fuláñez";  
print "<p>El nombre es $_SESSION[nombre]</p>";
```

Los dos programas siguientes acceden después a las matrices \$_SESSION, pero cada uno accede a la matriz correspondiente al nombre de sesión.

```
<?php  
session_name("ejemplo1");  
session_start();  
  
print "<p>El nombre es $_SESSION[nombre]</p>";
```

El nombre es Pepito Conejo

```
<?php  
session_name("ejemplo2");  
session_start();  
  
print "<p>El nombre es $_SESSION[nombre]</p>";
```

El nombre es Juan Fuláñez

El nombre de sesión distingue entre minúsculas y mayúsculas, es decir, dos sesiones con el mismo nombre, pero uno en minúsculas y otro en mayúsculas, son dos sesiones distintas.

El nombre de la sesión no puede contener únicamente números, ni tampoco puede contener los caracteres espacio (), punto (.), ampersand (&), más (+), corchete izquierdo ([) ni almohadilla (#).

Nota: No he encontrado una lista oficial de caracteres no admitidos. Los caracteres . & + [y # los he encontrado probando manualmente, pero es posible que haya otros caracteres problemáticos.

Referencia:

- [Caracteres admitidos en el identificador de sesión](#), que no es lo mismo que el nombre de sesión.

Borrar elementos de la sesión

Los valores de \$_SESSION se borran como en cualquier otra matriz mediante la función [unset\(\)](#).

```
<?php
session_start();

$_SESSION["nombre"] = "Pepito Conejo";

print "<p>Su nombre es $_SESSION[nombre].</p>\n";

unset($_SESSION["nombre"]);

if (isset($_SESSION["nombre"])) {
    print "<p>Su nombre es $_SESSION[nombre].</p>\n";
} else {
    print "<p>No sé su nombre.</p>\n";
}
```

```
<p>Su nombre es Pepito Conejo.</p>
<p>No sé su nombre.</p>
```

Para borrar todos los valores de `$_SESSION` se pueden borrar uno a uno o utilizar la función [session_unset\(\)](#), pero normalmente no se debe utilizar `unset()`:

- `unset($_SESSION)` impide que el resto de la página escriba o lea valores en `$_SESSION`, pero la sesión conserva los valores, por lo que otras páginas seguirían viendo esos valores.



```
<?php
session_start();
$_SESSION["nombre"] = "Pepito";
unset($_SESSION);
$_SESSION["apellidos"] = "Conejo";
header("Location:pagina-2.php");
```

```
<?php
session_start();
$_SESSION["saludo"] = "Hola";
print "<pre>\n";
print_r ($_SESSION);
print "</pre>\n";
```

```
<pre>
Array
(
    [nombre] => Pepito
    [saludo] => Hola
)
</pre>
```


- la función `session_unset()` borra todos los valores pero permite que el resto de la página (y otras páginas) escriba o lea valores en `$_SESSION`.



```
<?php
session_start();
$_SESSION["nombre"] = "Pepito";
session_unset();
$_SESSION["apellidos"] = "Conejo";
header("Location:pagina-2.php");
```

```
<?php
session_start();
$_SESSION["saludo"] = "Hola";
print "<pre>\n";
print_r ($_SESSION);
print "</pre>\n";
```

```
<pre>
Array
(
    [apellidos] => Conejo
    [saludo] => Hola
)
</pre>
```

Implementación de las sesiones

Al crear una sesión, el servidor asigna un identificador al usuario. Este identificador se guarda en el ordenador del cliente en forma de *cookie*, mientras que en el servidor se guarda un fichero con los valores de las variables de sesión. Cada vez que el cliente solicita una página al servidor, le envía la *cookie* que contiene el identificador y así el servidor puede recuperar los valores de la variable de sesión para utilizarlos al ejecutar la página.

De esta manera, el ordenador cliente no tiene acceso a los valores almacenados en la sesión.

- La página siguiente crea una variable de sesión

```
<?php
session_name("prueba");
session_start();

$_SESSION["nombre"] = "Barto";

print "<h1>Prueba de sesión</h1>\n";
```

- Al ejecutar la páginas, en el ordenador cliente se crea una *cookie*, como puede comprobarse con las herramientas de desarrollador web del navegador. En Firefox las *cookies* se pueden consultar en la pestaña Almacenamiento > Cookies. En Google Chrome las *cookies* se pueden consultar en la pestaña Aplicación > Almacenamiento > Cookies.

En la captura siguiente, se puede comprobar como el nombre de la sesión corresponde al nombre de la *cookie*, cuyo valor es el identificador de la sesión en el servidor.

The screenshot shows a web browser window with the address bar displaying `localhost/iaw/php/sesion.php`. The page title is **Prueba de sesión**. The developer tools are open, and the 'Almacenamiento' (Storage) tab is selected. In the left sidebar, the 'Cookies' section is expanded, showing a cookie for `http://localhost`. The main table lists the cookie with the following details:

Nombre	Valor	Domain	Path	Expires / Max-Age	Tamaño	HttpOnly
prueba	v50pjru619834...	localhost	/	Sesión	32	false

The 'Datos' (Details) panel on the right shows the following information for the selected cookie:

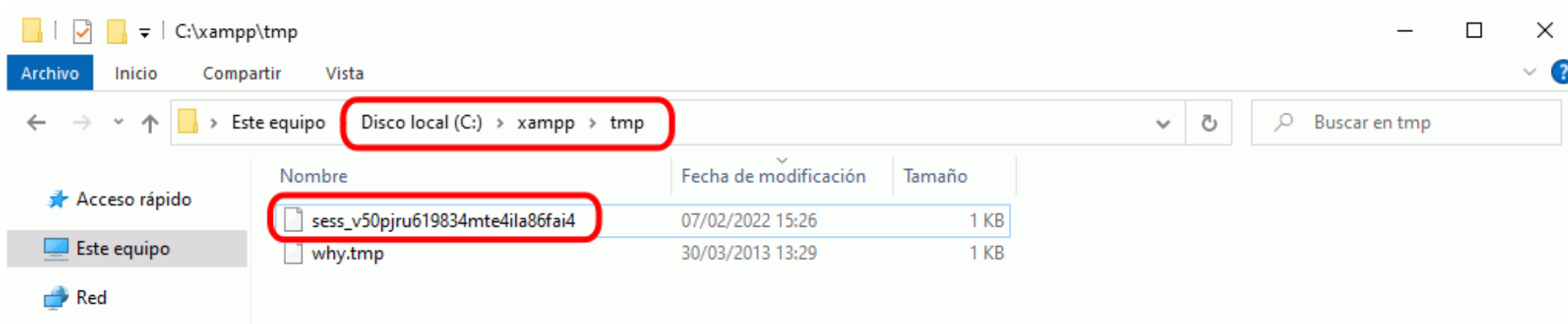
- prueba:** "v50pjru619834mte4ila86fai4"
- Creado:** "Mon, 07 Feb 2022 14:07:00 GMT"
- Domain:** "localhost"
- Expires / Max-Age:** "Sesión"
- HostOnly:** true
- HttpOnly:** false
- Path:** "/"

- En el servidor se guarda la información de las sesiones en ficheros en directorios cuya ubicación se puede establecer con la directiva de PHP `session.save_path` o modificar en una página con la función [`session_save_path\(\)`](#).



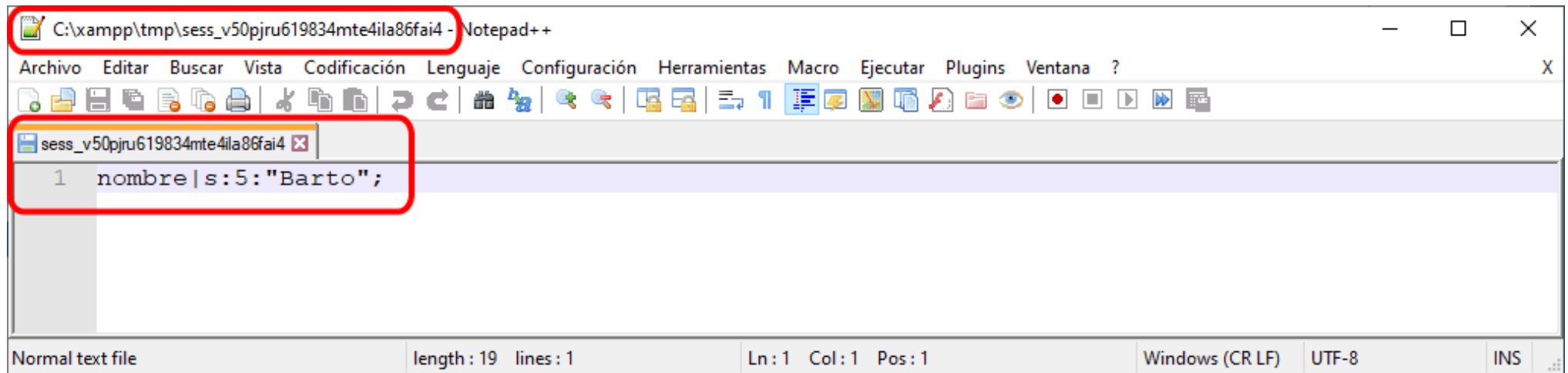
Directive	Local Value	Master Value
session.lazy_write	On	On
session.name	PHPSESSID	PHPSESSID
session.referer_check	no value	no value
session.save_handler	files	files
session.save_path	C:\xampp\tmp	C:\xampp\tmp
session.serialize_handler	php	php
session.sid_bits_per_character	5	5
session.sid_length	26	26

- En ese directorio podemos encontrar ficheros de texto plano que incluyen en su nombre el identificador de la sesión.



Nombre	Fecha de modificación	Tamaño
sess_v50pjru619834mte4ila86fai4	07/02/2022 15:26	1 KB
why.tmp	30/03/2013 13:29	1 KB

- El fichero contiene los nombres y valores de las variables de sesión:



Directiva session.save_handler

Para utilizar sesiones mediante el mecanismo propio de PHP (es decir, sin necesidad de crear funciones propias), la directiva `session.save_handler` del archivo de configuración `php.ini` debe tener el valor `files`.

```
session.save_handler = files ; Valor recomendado en este curso
```

Esta configuración es la más habitual, pero algunos gestores de contenidos (CMS) tienen sus propias funciones de gestión de sesiones y requieren que esta directiva tome el valor `user`.

```
session.save_handler = user ; Valor definido en algunos servidores
```

Si se ha necesitado modificar `php.ini`, pero queremos ejecutar otros programas que no incluyen sus propias funciones de gestión de sesiones, se puede incluir en las páginas PHP la función `ini_set()` antes de abrir la sesión. Ese cambio sólo afectará a la página que incluya la llamada a la función.

```
ini_set("session.save_handler", "files"); // Necesario únicamente cuando session.save_handler = user en php.ini
session_start();
```

En este curso se supondrá que la directiva tiene el valor `"files"`.



Esta página forma parte del curso [Programación web en PHP](#) por [Bartolomé Sintes Marco](#) que se distribuye bajo una [Licencia Creative Commons Reconocimiento-CompartirIgual 4.0 Internacional \(CC BY-SA 4.0\)](#).