



Code > PHP

Cómo Usar Sesiones y Variables de Sesión en PHP

**Sajal Soni** Feb 16, 2021

Read Time: 8 mins



Español



PHP

Back-End

Cookies

Language Fundamentals

Spanish (Español) translation by [Víctor Ponz Artero](#) (you can also [view the original English article](#))

El manejo de sesiones es un concepto clave en PHP que permite que la información de usuario persista entre todas las páginas de un sitio web o app. En este post, aprenderás los fundamentos para manejar sesiones en PHP.

Empezaremos con una explicación de cómo funcionan las sesiones y cómo estas están relacionadas con las cookies. Después daremos un vistazo a varios fragmentos de código que demuestran cómo trabajar con sesiones. Aprenderás a crear y destruir sesiones, y a cambiar variables de sesión.

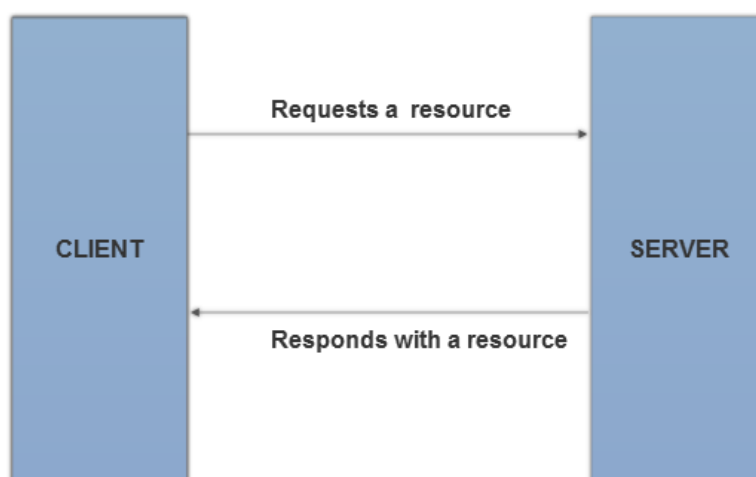


¿Qué es una Sesión en PHP?

Una sesión es un mecanismo para persistir información en diferentes páginas web para identificar usuarios mientras estos navegan un sitio o app. Te preguntarás por qué las sesiones son necesarias en un sitio web. Para ver porqué las sesiones son necesarias, tenemos que viajar atrás y ver como esta diseñado el protocolo HTTP.

El protocolo HTTP es un protocolo sin estado, lo que significa que no hay forma de que un servidor recuerde a un usuario específico entre múltiples peticiones. Por ejemplo, cuando accedes a una página web, el servidor sólo es responsable de proveer el contenido de la página solicitada. Así que cuando accedes a otras páginas en el mismo sitio web, el servidor web interpreta cada petición separadamente, como si no estuvieran relacionadas unas con otras. No hay forma para el servidor de sane que cada petición fue originada por el mismo usuario.

El siguiente diagrama refleja el protocolo HTTP en pocas palabras.



En este modelo, si quieres mostrar información relativa a un usuario específico deberás autenticar al usuario en cada petición. ¡Imagina que tuvieras que



introducir tu nombre de usuario y contraseña en cada página que muestre tu información de perfil! Sí, sería incómodo y nada práctico, y aquí es donde las sesiones entran en juego.

Una sesión permite compartir información entre las diferentes páginas de un único sitio web o app, así que ayuda a mantener el estado. Esto permite al servidor conocer que todas las peticiones se originan desde el mismo usuario, permitiendo al sitio web mostrar información y preferencias específicas de ese usuario.

Flujo de login con Sesiones y Cookies

Vamos a ver rápidamente un flujo de login común para un sitio web para entender qué ocurre entre bambalinas.

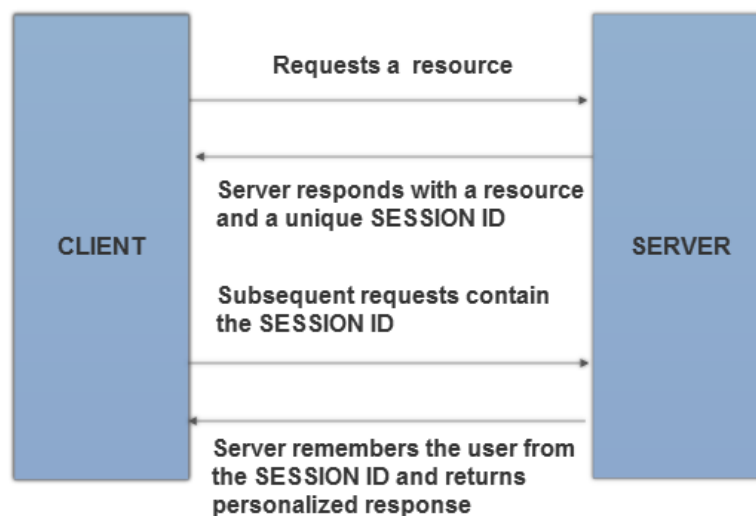
1. Un usuario accede a la página de login de un sitio web.
2. Después de enviar el formulario de login, un servidor en el otro extremo autentica la petición revalidando las credenciales que fueron introducidas.
3. Si las credenciales introducidas por el usuario son válidas, el servidor crea una nueva sesión. El servidor genera un número aleatorio único, que es llamado identificador de sesión (session id en inglés). También crea un nuevo fichero en el servidor que es usado para almacenar información específica para dicha sesión.
4. A continuación, el identificador de sesión es enviado de vuelta al usuario, junto con cualquier recurso que este hubiera solicitado. Entre bambalinas, el identificador de sesión es enviado en la cookie `PHPSESSID` en la cabecera de respuesta.
5. Cuando el navegador recibe la respuesta del servidor, aparece el encabezado de la cookie `PHPSESSID`. Si el navegador permite las cookies, guardará esta cookie `PHPSESSID`, que almacena el identificador de sesión que el servidor ha pasado.
6. En las subsecuentes peticiones, la cookie `PHPSESSID` es devuelta al servidor. Cuando el servidor recibe la cookie `PHPSESSID`, intentará inicializar una sesión con dicho identificador de sesión. Esto lo hace cargando el fichero de sesión que fue creado previamente durante la inicialización de la sesión. Después inicializa la variable `$_SESSION`, que es



un array super-global, con los datos almacenados en dicho fichero de sesión.

De esa forma, los datos de usuario son preservados entre múltiples peticiones y este se mantiene registrado durante toda la sesión.

El siguiente diagrama refleja cómo el protocolo HTTP trabaja con sesiones.



Ahora que has visto una breve introducción a cómo funcionan las sesiones, crearemos unos cuantos ejemplos prácticos para demostrar cómo crear y manipular variables de sesión.

¿Cómo iniciar una Sesión?

En esta sección, discutiremos cómo iniciar una sesión en PHP.

Cuando quieras tratar con variables de sesión, necesitas asegurarte de que la sesión ya haya empezado. Hay varias formas de iniciar una sesión en PHP.

Usa la Función `session_start`.



Este será el método que verás más a menudo, cuando la sesión es iniciada mediante la función `session_start`.

```
1 <?php
2 // start a session
3 session_start();
4
5 // manipulate session variables
6 ?>
```

Lo importante es que la función `session_start` debe ser llamada al principio del script, antes de enviar cualquier salida al navegador. De otra forma, encontrarás el infame error `Headers are already sent`.

Advertisement

Iniciar una Sesión automáticamente

Si existe la necesidad de usar sesiones a lo largo de toda tu aplicación, también puedes optar por iniciar la sesión automáticamente sin necesidad de usar la función `session_start`.

Hay una opción en el archivo de configuración **php.ini** que te permite iniciar una sesión automáticamente para cada petición — `session.auto_start`. Por defecto, está fijada a `0`, pero puedes fijarla a `1` para activar la función de aut
inicio



```
ini_file;
```

```
1 session.auto_start = 1
```

Por otro lado, si no tienes acceso al fichero **php.ini**, y estás usando el servidor web Apache, puedes fijar esta variable usando el fichero **.htaccess**.

```
1 php_value session.auto_start 1
```

Si añades la línea anterior al fichero **.htaccess**, iniciará una sesión automáticamente en tu aplicación PHP.

¿Cómo obtener un identificador de Sesión?

Como discutimos anteriormente, el servidor crea un número único para cada nueva sesión. Si quieres obtener el identificador de sesión, puedes usar la función `session_id`, como muestra el siguiente fragmento de código:

```
1 <?php
2 session_start();
3 echo session_id();
4 ?>
```

Esto debería darte el identificador de sesión actual. La función `session_id` es interesante en porque también puede recibir un argumento— un identificador de sesión. Si quieres reemplazar el identificador de sesión generado por el sistema por el tuyo propio, puedes suministrarlo como el primer argumento de la función `session_id`.

```
1 <?php
2 session_id(YOUR_SESSION_ID);
3 session_start();
4 ?>
```

Es importante hacer notar que la función `session_id` debe estar situada antes que la llamada a `session_start` cuando quieras iniciar una sesión con un identificador de sesión personalizado.



¿Cómo Crear Variables de Sesión?

En esta sección, exploraremos cómo inicializar variables de sesión en PHP.

Como discutimos anteriormente, una vez que una sesión es iniciada, el array super-global `$_SESSION` es inicializado con la correspondiente información de sesión. Por defecto, se inicializa con un array vacío, y puedes almacenar más información usando un par clave-valor.

Veamos el siguiente script de ejemplo que muestra cómo inicializar las variables de sesión.

```
01 <?php
02 // start a session
03 session_start();
04
05 // initialize session variables
06 $_SESSION['logged_in_user_id'] = '1';
07 $_SESSION['logged_in_user_name'] = 'Tutsplus';
08
09 // access session variables
10 echo $_SESSION['logged_in_user_id'];
11 echo $_SESSION['logged_in_user_name'];
12 ?>
```

Como puedes ver, hemos iniciado una sesión al principio del script usando la función `session_start`. A continuación, hemos inicializado un par de variables de sesión. Finalmente, hemos accedido a dichas variables usando la super-global `$_SESSION`.

Cuando almacenas datos en una sesión usando la super-global `$_SESSION`, finalmente se almacenan en su correspondiente fichero de sesión en el servidor que fue creado cuando la sesión fue iniciada. De esta forma, los datos de sesión son compartidos entre múltiples peticiones.



Como discutimos, la información de sesión se comparte entre peticiones, y de

esta forma las variables de sesión inicializadas en una página pueden ser accedidas desde otras páginas también, hasta que la sesión expira. Generalmente, una sesión expira cuando se cierra el navegador.

Advertisement

¿Cómo Modificar y Borrar Variables de Sesión?

Puedes modificar y borrar variables creadas previamente en la aplicación de la misma manera que para las variables PHP regulares.

Veamos cómo modificar variables de sesión.

```
01 <?php
02 session_start();
03
04 if (!isset($_SESSION['count']))
05 {
06     $_SESSION['count'] = 1;
07 }
08 else
09 {
10     ++$_SESSION['count'];
11 }
12
13 echo $_SESSION['count'];
14 ?>
```



En el script anterior, hemos comprobado en primer lugar si la variable `$_SESSION['count']` está fijada. Si no lo está, la fijamos a `1`, en otro caso la incrementamos en `1`. Así que si refrescas la página múltiples veces, deberías ver que ¡el contador se incrementa en uno cada vez!

Por otro lado, si desearas borrar una variable de sesión, puedes usar la función `unset`, como se muestra en el siguiente fragmento de código.

```
01 <?php
02 // start a session
03 session_start();
04
05 // initialize a session variable
06 $_SESSION['logged_in_user_id'] = '1';
07
08 // unset a session variable
09 unset($_SESSION['logged_in_user_id']);
10 ?>
```

Por lo tanto, ya no puedes acceder a la variables `$_SESSION['logged_in_user_id']` ya que se eliminó por la función `unset`. Así es como puedes alterar la información de la sesión.

¿Cómo Destruir una Sesión?

En esta sección, veremos cómo puedes destruir una sesión. En la sección anterior, discutimos la función `unset`, que se usa si quieres eliminar variables de sesión específicas. Por otro lado, si quieres eliminar toda la información relacionada con la sesión, puedes usar la función `session_destroy`.

Intentemos entender cómo funciona usando el siguiente ejemplo.

```
1 <?php
2 // start a session
3 session_start();
4
5 // assume that we've initialized a couple of session variables in the
```



```
6  
7 // destroy everything in this session  
8 session_destroy();  
9 ?>
```

La función `session_destroy` elimina todo lo que se almacena en la sesión actual. Por lo tanto, verás la variable `$_SESSION` vacía en las subsecuentes peticiones ya que la función `session_destroy` eliminó todos los datos de sesión almacenados en disco.

Generalmente, usarás la función `session_destroy` cuando el usuario va a ser desconectado.

Conclusión

En este artículo, hemos explorado los fundamentos del tratamiento de sesiones en PHP. Es un concepto clave que permite persistir información entre páginas web.

En la primera parte del artículo, hemos discutido los conceptos básicos de las sesiones, y más tarde creamos varios ejemplos en PHP para demostrar cómo puedes crear y destruir sesiones así como manipular variables de sesión.

Did you find this post useful?



Yes



No

Want a weekly email summary?

Subscribe below and we'll send you a weekly email summary of all new Code tutorials. Never miss out on learning about the next big thing.

Sign up





Sajal Soni

Software Engineer, FSPL, India

I'm a software engineer by profession, and I've done my engineering in computer science. It's been around 14 years I've been working in the field of website development and open-source technologies.

Primarily, I work on PHP and MySQL-based projects and frameworks. Among them, I've worked on web frameworks like CodeIgnitor, Symfony, and Laravel. Apart from that, I've also had the chance to work on different CMS systems like Joomla, Drupal, and WordPress, and e-commerce systems like Magento, OpenCart, WooCommerce, and Drupal Commerce.

I also like to attend community tech conferences, and as a part of that, I attended the 2016 Joomla World Conference held in Bangalore (India) and 2018 DrupalCon which was held in Mumbai (India). Apart from this, I like to travel, explore new places, and listen to music!

 [sajalsoni](#)

 FEED  LIKE  FOLLOW



Advertisement

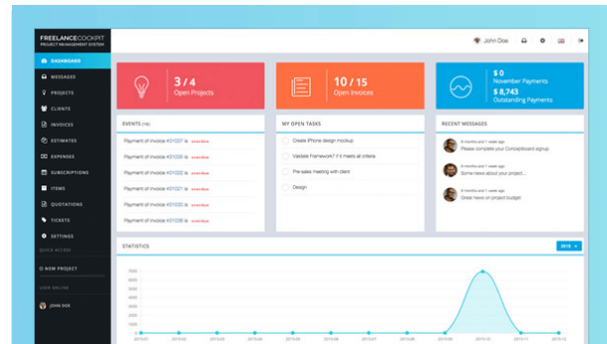
LOOKING FOR SOMETHING TO HELP KICK START YOUR NEXT PROJECT?

Envato Market has a range of items for sale to help get you started.



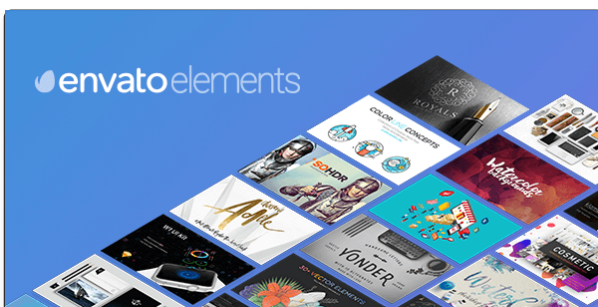
WordPress Plugins

From \$5



PHP Scripts

From \$5



Unlimited Downloads
From \$16.50/month

Get access to over one million creative assets on Envato Elements.



Over 9 Million Digital Assets

Everything you need for your next creative project.



QUICK LINKS - Explore popular categories**ENVATO TUTS+**

About Envato Tuts+
Terms of Use
Advertise

JOIN OUR COMMUNITY

Teach at Envato Tuts+
Translate for Envato Tuts+
Forums

HELP

FAQ
Help Center

**tuts+**

30,800
Tutorials

1,316
Courses

50,290
Translations



Envato Envato Elements Envato Market Placeit by Envato Milkshake All

products Careers Sitemap

© 2022 Envato Pty Ltd. Trademarks and brands are the property of their respective owners.

