Bloque B

Páginas Web Dinámicas

Unidad 9. Cookies & Sesiones

Contenidos

1. Introducción

Introducción

Para crear páginas web que contengan datos personales como un nombre de usuario, una foto de perfil o una lista de páginas vistas recientemente, **el sitio necesitaría saber quién solicita cada página**.

El protocolo HTTP proporciona reglas que especifican cómo debe solicitar un navegador una página web y cómo debe responder el servidor, pero **trata cada solicitud y respuesta por separado**. HTTP **no proporciona un mecanismo para que un sitio web sepa qué visitante está solicitando una página**.

Introducción

Si un sitio necesita saber quién solicita una página web o mostrar información personalizada, puede hacer un **seguimiento de cada visitante** y **almacenar información sobre sus preferencias** utilizando una **combinación de cookies y sesiones**.

- Las **cookies** son archivos de texto que se almacenan en el navegador del usuario. Un sitio puede indicar al navegador qué datos almacenar en una cookie, y el navegador devolverá esos datos al sitio web con cada página posterior que solicite de ese sitio.
- Las **sesiones** permiten a un sitio almacenar temporalmente datos sobre un usuario en el servidor. Cuando un visitante solicita otra página del sitio, el intérprete de PHP puede acceder a los datos de la sesión de ese usuario.

Introducción

Las cookies y las sesiones **almacenan pequeñas cantidades de datos temporalmente**, pero no se garantiza que almacenen datos durante un largo periodo de tiempo porque los usuarios pueden borrar las cookies (o acceder al sitio desde un navegador diferente que no tenga la cookie), y las sesiones sólo están diseñadas para durar lo que dura una visita al sitio (no almacenan datos entre visitas).

Cuando los datos sobre los usuarios deben almacenarse durante **períodos más largos**, **se guardan en una base de datos**. Aprenderás cómo hacerlo en la unidad 13. Para ello es necesario saber cómo funcionan las cookies y las sesiones.

2. Cookies

Cookies

Un sitio web puede indicar a un navegador que almacene datos sobre el usuario en un archivo de texto llamado **cookie**. Entonces, cada vez que el navegador solicita otra página de ese sitio, envía los datos de la cookie al servidor.

Cookies

¿Que es una cookie?

Un sitio web puede indicar a un navegador que cree una cookie, que es un archivo de texto que se almacena en el navegador.

Cada cookie tiene un nombre, que debe describir el tipo de información que contiene. El nombre de la cookie será el mismo para cada visitante.

El valor almacenado en la cookie de cada usuario puede cambiar. Así que **una cookie es como una variable que se almacena en un archivo de texto en el navegador del usuario**.

Cookies

Creando cookies

Cuando un navegador solicita una página web, el sitio web puede enviar una cabecera HTTP adicional de vuelta al navegador con la página.

Esa cabecera HTTP indica al navegador el nombre de la cookie que debe crear y el valor que debe almacenarse en la cookie.

El valor almacenado en la cookie es **texto**, y no debe tener más de 4.096 caracteres. **Un sitio también puede crear más de una cookie**.

Cookies

Obteniendo datos de las cookies

Si el navegador solicita otra página del sitio que creó la cookie, enviará el nombre de la cookie y el valor que almacena al servidor junto con la solicitud de página.

El intérprete de PHP añade entonces los datos de la cookie a un array superglobal llamado `$_COOKIE` para que el código PHP de esa página pueda utilizarlos. **El nombre de la cookie es la clave, y su valor es el valor que la cookie almacenó**.

Cookies

¿Quién puede acceder a una cookie?

Los navegadores sólo envían datos en una cookie a un servidor cuando éste solicita una página del mismo dominio que la creó.

Por ejemplo, si google.com crea una cookie, sólo se envía cuando el navegador solicita páginas de google.com. Nunca se enviaría a facebook.com.

> JavaScript, en general, también puede acceder a los datos de las cookies.

Cookies

Las cookies están vinculadas a un navegador

Porque es el navegador el que crea y almacena las cookies:

- Si hay más de un navegador instalado en un dispositivo, **la cookie sólo se envía desde el navegador en el que se almacenó** (no desde cualquier otro navegador instalado en ese dispositivo).
- Si un usuario consigue un nuevo dispositivo, ese dispositivo **no tendrá la cookie para enviar al servidor**.

Cookies

¿Cuánto duran las cookies?

El servidor puede **especificar una fecha y hora en la que expira una cookie**. Esta es la fecha y hora en la que el navegador debe dejar de enviar los datos de la cookie al servidor. Si no se indica una fecha de caducidad, **el navegador deja de enviar la cookie al servidor cuando el usuario cierra su navegador**.

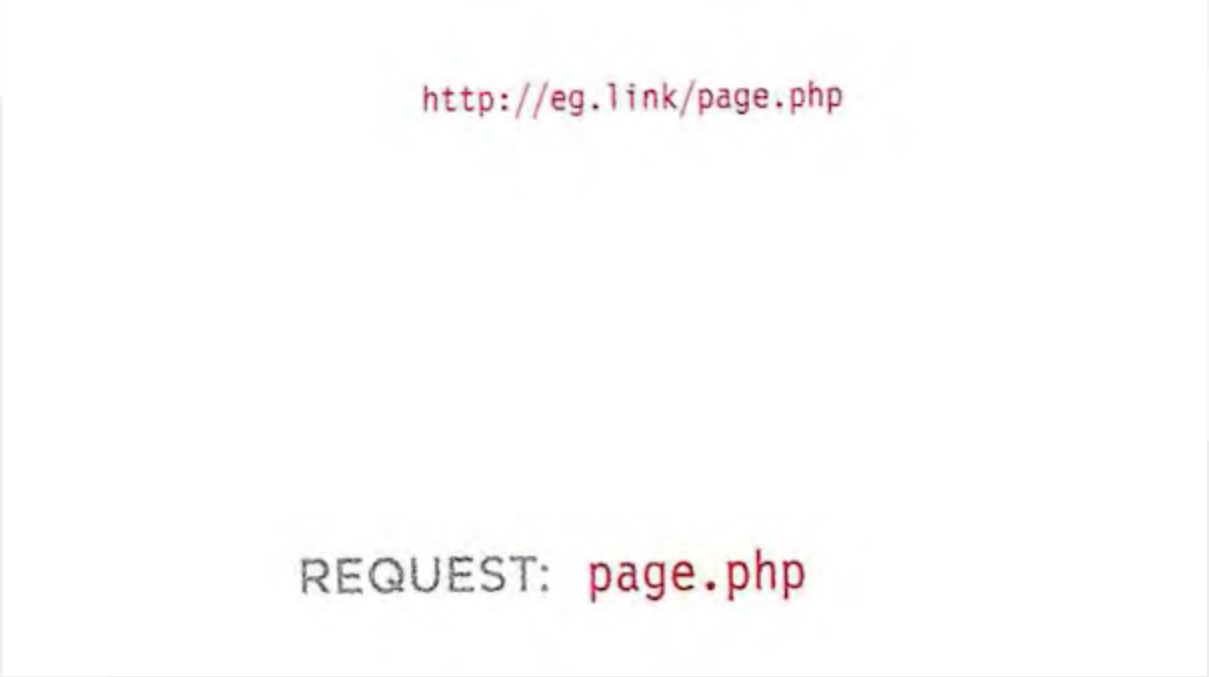
> Los usuarios también pueden rechazar o eliminar las cookies, por lo que un sitio debería poder funcionar sin ellas.

Cookies

Ejemplo de peticiones usando Cookies

Primera solicitud de la página

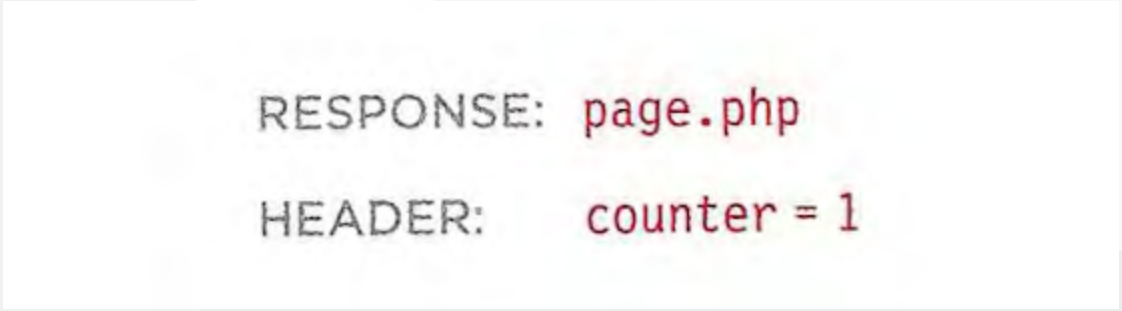
- El navegador solicita una página utilizando una petición HTTP



Cookies

Primera solicitud de la página

- El servidor devuelve la página solicitada
- Añade una cabecera HTTP que indica al navegador el nombre de la cookie que debe crear y su valor.



Cookies

Primera solicitud de la página

- El navegador muestra la página
- Crea una cookie utilizando los datos de la cabecera HTTP

Cookies

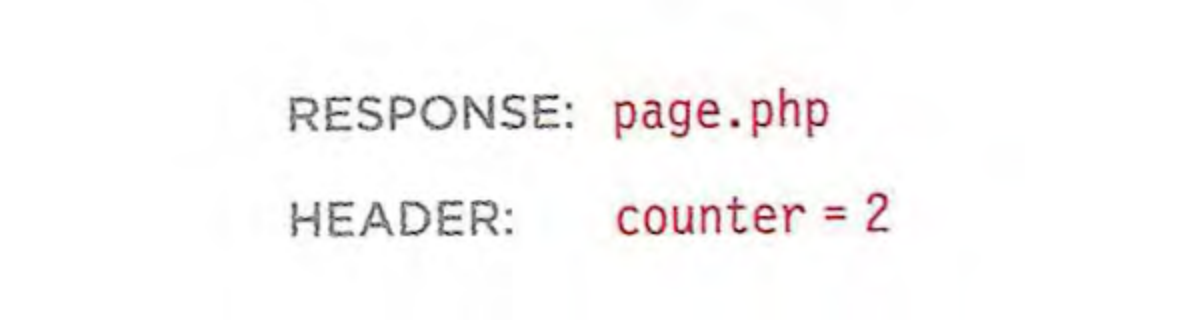
Peticiones de página posteriores

- El navegador solicita una página utilizando una petición HTTP
- Envía la cabecera HTTP con el nombre y el valor de la cookie

Cookies

Peticiones de página posteriores

- El servidor añade los datos de la cookie a `$_COOKIE`
- Crea la página utilizando los datos de `$_COOKIE`
- Devuelve la página solicitada
- Puede actualizar el valor almacenado en la cookie



Cookies

Peticiones de página posteriores

- El navegador muestra la página creada utilizando los datos de su cookie
- Actualiza la cookie utilizando los datos de la cabecera HTTP

Cookies

Una cookie **no debe utilizarse para almacenar datos confidenciales** (por ejemplo, direcciones de correo electrónico o números de tarjetas de crédito) porque el contenido de la cookie puede verse en las herramientas de desarrollo de un navegador y se envía entre un navegador y un servidor como texto sin formato.

Para evitar que alguien lea las cabeceras HTTP cuando se envían entre el navegador y el servidor hay que ejecutar nuestro sitio utilizando **HTTPS** en lugar de HTTP (ver *introducción del bloque B*). Esto **encripta la información de las cabeceras**.

Cómo crear y acceder a cookies

La función incorporada `setcookie()` de PHP se utiliza para **crear una cookie**. Para acceder a las cookies se puede utilizar el array superglobal `$_COOKIE`, o las funciones `filter_input()` y `filter_input_array()`.

Cómo crear y acceder a cookies

La función `setcookie()` de PHP **crea una cabecera HTTP que se envía con la página web y le indica al navegador que cree una cookie**. La función permite establecer un **nombre** y un **valor** para la cookie.

Cómo crear y acceder a cookies

Como esta función crea una cabecera HTTP, debe utilizarse **antes de enviar el contenido al navegador** (como se mostró con la función `header()` en la unidad 5). Incluso un espacio antes de la etiqueta `<?php` de apertura se trata como contenido.

Si no se establece una fecha de caducidad para una cookie, el navegador deja de enviar los datos de la cookie al servidor **cuando el usuario cierra su navegador**. Más adelante veremos cómo establecer una fecha de caducidad para una cookie.

```php
setcookie($name, $value);
```

Cómo crear y acceder a cookies

Una vez que el navegador ha almacenado la cookie, si ese navegador solicita otra página del sitio, **el nombre y el valor de la cookie se envían al servidor con la solicitud**. Cuando el intérprete de PHP recibe la petición, **añade los datos de la cookie a un array superglobal** llamado `$_COOKIE` .

Cómo crear y acceder a cookies

Se añade un nuevo elemento al array por cada cookie. Para cada elemento:

- La **clave** es el nombre de la cookie
- El **valor** es el dato que contiene la cookie (se almacena como una cadena)

Estos datos **suelen recogerse y almacenarse en una variable**.

Cómo crear y acceder a cookies

Si el código intenta acceder a una clave que no existe en `$_COOKIE` se genera un error. Para evitar esto, se puede utilizar el **operador de coalescencia nula** para comprobar si la clave está en el array. Si lo está, el valor de la cookie se almacena en la variable. Si no, la variable almacena el valor *null*.

```php
$preference = $_COOKIE['name'] ?? null;
```

Cómo crear y acceder a cookies

Las funciones `filter_input()` y `filter_input_array()` de PHP (ver unidad 6) también pueden recopilar datos de cookies. El tipo de entrada debe establecerse como *INPUT_COOKIE*.

El segundo parámetro es el **nombre de la cookie**. El tercer y cuarto parámetro son opcionales; especifican el ID del filtro a utilizar y cualquier opción para el filtro.

Si no se ha enviado una cookie, la función no genera ningún error. Además, si se utilizan filtros de tipo entero, flotante o booleano, convierten el valor a ese tipo de datos.

```
$preference = filter_input(INPUT_COOKIE, $name[, $filter[, $options]]);
```

Ejemplo: Establecer y acceder a cookies

Este ejemplo utiliza una cookie para contar el número de páginas que ha visto un visitante.

1. La variable `$counter` almacena el número de páginas que ha visto el visitante. Si el navegador envía datos de una cookie llamada counter al servidor, `$counter` almacenará ese valor. Si no, se utiliza el operador null-coalescing para almacenar un valor de 0 en su lugar.
2. Se añade 1 al valor en `$counter`, ya que el visitante acaba de ver una página.
3. La función `setcookie()` se utiliza para indicar al navegador que cree o actualice una cookie llamada *counter*, y almacene el valor de `$counter` en esa cookie.

Ejemplo: Establecer y acceder a cookies

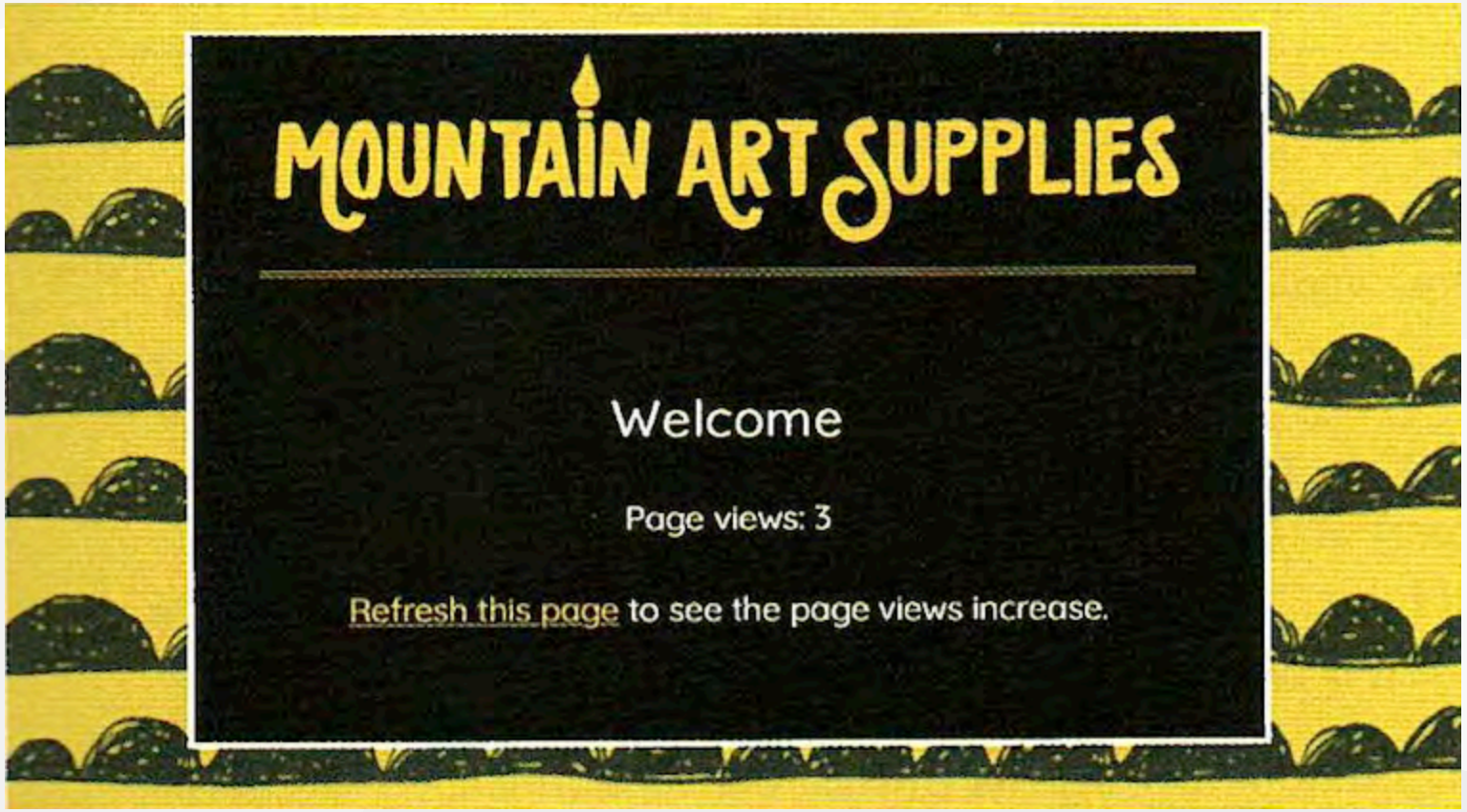
4. La variable `$message` almacena un mensaje que dice el número de páginas que ha visto el visitante.
5. Se muestra el mensaje.

> PRUEBA: Una vez que haya visto la página una vez, actualiza la página y observa cómo sube el contador.
> PRUEBA: Almacena tu nombre en una cookie llamada nombre y muéstrala después de las visitas a la página.

Ejemplo: Establecer y acceder a cookies

```php
<?php
① $counter = $_COOKIE['counter'] ?? 0;   // Get data
② $counter = $counter + 1;               // +1 to counter
③ setcookie('counter', $counter);        // Update cookie

④ $message = 'Page views: ' . $counter; // Message
?>
<?php include 'includes/header.php'; ?>

<h1>Welcome</h1>
⑤ <p><?= $message ?></p>
<p><a href="sessions.php">Refresh this page</a> to see
the page views increase.</p>

<?php include 'includes/footer.php'; ?>
```

Ejemplo: Establecer y acceder a cookies



Actividad: Establecer y acceder a cookies

Imagina que trabajas en un sitio web donde se quiere personalizar la bienvenida de los usuarios, recordando su nombre durante la sesión del navegador actual. Cada vez que el usuario visite la página durante esa sesión, verá un mensaje de bienvenida personalizado.

Actividad: Establecer y acceder a cookies

Instrucciones

1. **Crea un formulario** en `welcome.php` que pida el nombre del usuario si no se ha almacenado previamente en una cookie. Si el nombre ya está en la cookie, dale la bienvenida automáticamente.
2. **Almacena el nombre en una cookie** al enviar el formulario sin definir un tiempo de expiración, para que la cookie se elimine cuando el usuario cierre el navegador.

Actividad: Establecer y acceder a cookies

3. **Muestra un mensaje de bienvenida** en función de la cookie:
    - Si el usuario ya tiene una cookie de nombre, muestra: "Bienvenido de nuevo, [nombre]".
    - Si el usuario es nuevo (sin cookie), muestra un formulario que pida su nombre.

Controlando la configuración de las Cookies

`setcookie()` tiene parámetros que controlan cómo utilizan las cookies los navegadores. También es conveniente validar los datos recibidos de las cookies y utilizar `htmlspecialchars()` si su contenido se muestra en una página.

Controlando la configuración de las Cookies

- Para **actualizar un valor almacenado en una cookie**, hay que llamar de nuevo a `setcookie()` con un nuevo valor para la cookie.
- Para que el navegador **deje de enviar una cookie**, se debe volver a llamar a `setcookie()` estableciendo el valor a una cadena en blanco y la caducidad a un tiempo en el pasado.

Controlando la configuración de las Cookies

Siempre que actualicemos el valor o el tiempo de expiración de una cookie, los últimos cuatro argumentos deben utilizar los mismos valores que se utilizaron cuando se creó la cookie, lo que asegura que la cookie sea tratada de manera coherente en el navegador (si no se hace así, el navegador podría interpretar que estás trabajando con una cookie distinta, en lugar de actualizar o eliminar la existente).

```
setcookie($name[, $value, $expire, $path, $domain, $secure, $httponly])
```

Controlando la configuración de las Cookies

Además, debido a que es posible enviar cabeceras HTTP que imiten cookies con una petición de página:

- El servidor debería **validar los datos de las cookies antes de utilizarlos** (utilizando las técnicas estudiadas en la unidad 6).
- Si se muestra un valor de cookie en una página, se debería utilizar `htmlspecialchars()` para prevenir un ataque XSS.

Controlando la configuración de las Cookies

PARAMETER	DESCRIPTION
$name	The name of the cookie.
$value	The value the cookie should hold (this gets treated as a string - cookies do not store data types).
$expire	The date and time the browser should stop sending the cookie to the server (as a Unix timestamp). To set the timestamp, use PHP's `time()` function and add the period you want the cookie to last for.
$path	If a cookie is only needed for part of the site, specify the directories it should be used for. By default, the path is the root folder '/' which means all directories. Setting this to `/members` means it is only sent to pages in the `members` folder of the site.
$domain	If the cookie is only needed on a subdomain, set the URL for the subdomain. By default, it is sent to all subdomains of a site. If it is set to the subdomain `members.example.org`, the cookie is only sent to files in the subdomain `members.example.org`.
$secure	If this is given a value of `true`, the cookie will be created in the browser, but the browser only sends it back to the server if the page is requested using a secure HTTPS connection (see p184).
$httponly	If given a value of `true`, the cookie is only sent to the server (it cannot be accessed by JavaScript).

PERIOD	NOW	SECS	MINS	HRS	DAYS
1 day	`time() +`	`60 *`	`60 *`	`24`	
30 days	`time() +`	`60 *`	`60 *`	`24 *`	`30`

Ejemplo: Controlando la configuración de las Cookies

Este ejemplo **permite al usuario seleccionar un esquema de color** ("dark" o "light") **y guarda la preferencia en una cookie**; al recargar la página, se aplica el esquema de color almacenado, o el esquema "dark" por defecto si no se ha definido ninguna preferencia válida.

Ejemplo: Controlando la configuración de las Cookies

1. La variable `$color` almacena el valor enviado para una cookie llamada color (o null si no se envió).
2. Un array almacena las opciones permitidas para la combinación de colores.
3. Una sentencia if comprueba si el formulario ha sido enviado.
4. Si lo ha sido, el valor de la caja de selección llamada color se almacena en la variable `$color`. Esto sobrescribe el valor del paso 1.

Ejemplo: Controlando la configuración de las Cookies

5. La función `setcookie()` es llamada para establecer una cookie llamada color. Su valor es la opción que el usuario seleccionó de la caja de selección. También:

- Caduca en una hora
- Se envía a todas las páginas del sitio
- Se envía a través de HTTP o HTTPS
- Se oculta a partir de JavaScript

6. La condición de un operador ternario comprueba si el valor en `$color` está en el array `$options`. Si lo está, el valor se guarda en la variable llamada `$scheme`. Si no lo está, `$scheme` guarda el valor dark.
7. Se incluye una nueva cabecera. Escribe el valor en la variable `$color` en el atributo class de la etiqueta `<body>` para asegurar que las reglas CSS de la página utilizan el esquema de color correcto.

Ejemplo: Controlando la configuración de las Cookies

cookie-preferences.php

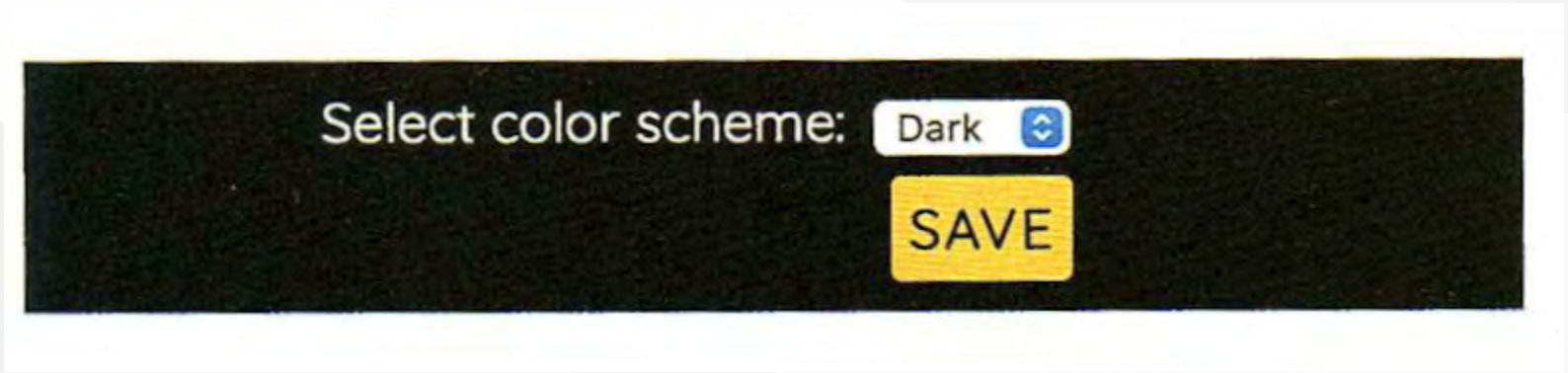
```
<?php
① $color   = $_COOKIE['color'] ?? null;        // Get data
② $options = ['light', 'dark',];                // Options

③ if ($_SERVER['REQUEST_METHOD'] == 'POST') { // If posted
④     $color = $_POST['color'];                 // Get color
⑤     setcookie('color', $color, time() + 60 * 60,
          '/', '', false, true);                // Set cookie
  }

  // If color is valid option, use it - otherwise use dark.
⑥ $scheme = (in_array($color, $options)) ? $color : 'dark';
  ?>
⑦ <?php include 'includes/header-style-switcher.php'; ?>
    <form method="POST" action="cookie-preferences.php">
      Select color scheme:
      <select name="color">
        <option value="dark">Dark</option>
        <option value="light">Light</option>
      </select><br>
      <input type="submit" value="Save">
    </form>
  <?php include 'includes/footer.php'; ?>
```

Ejemplo: Controlando la configuración de las Cookies

includes/header-style-switcher.php

```
<body class="<?= htmlspecialchars($scheme) ?>">
```

Ejemplo: Controlando la configuración de las Cookies



Actividad: Controlando la configuración de las Cookies

Imagina que trabajas en un proyecto de un sitio web multilingüe y quieres que la página recuerde la preferencia de idioma del usuario cada vez que regrese. De esta manera, al visitar el sitio, los usuarios no necesitan seleccionar su idioma cada vez.

Actividad: Controlando la configuración de las Cookies

Instrucciones

1. **Crear un formulario de selección de idioma** en `language-preferences.php` que permita al usuario seleccionar su idioma preferido de entre dos opciones: Inglés y Español.
2. **Almacenar la preferencia en una cookie** al enviar el formulario, con una duración de 30 días.
3. **Mostrar contenido en el idioma seleccionado**:
    - Si existe la cookie de idioma, muestra un mensaje de bienvenida en ese idioma.
    - Si no existe la cookie, muestra el formulario de selección de idioma.

3. Sesiones

Sesiones

Las **sesiones** almacenan información sobre un usuario y sus preferencias en el servidor. Se llaman sesiones porque **sólo almacenan los datos durante una única visita al sitio**.

Sesiones

¿Qué es una sesión?

Cuando se inicia una sesión, el intérprete de PHP crea tres cosas:

- Un **identificador de sesión** (session ID), una cadena utilizada para identificar a un visitante individual.
- Un **archivo de sesión** (session file), que es un archivo de texto que se almacena en el servidor. Se utiliza para guardar datos sobre ese usuario. Su nombre contendrá el ID de sesión.
- Una **cookie de sesión** (session cookie), que se almacena en el navegador. Su nombre es `PHPSESSID` y su valor es el ID de sesión de ese usuario.

Sesiones

Obtención de datos de sesión

Si un navegador tiene una cookie de sesión, ésta se envía al servidor cada vez que el usuario solicita otra página de ese sitio.

El ID de sesión se utiliza para identificar al usuario, de modo que el servidor pueda:

- Encontrar el archivo de sesión cuyo nombre de archivo contiene el ID de sesión enviado en la cookie.
- Tomar los datos del archivo de sesión y ponerlos en el array superglobal $_SESSION para que la página pueda acceder a ellos.

Sesiones

Guardar datos de sesión

Una vez que se ha creado una sesión, se pueden guardar nuevos datos en la sesión de ese usuario añadiéndolos al array superglobal `$_SESSION`.

Cuando una página ha terminado de ejecutarse, el intérprete de PHP toma todos los datos del array superglobal `$_SESSION` y los guarda en el archivo de sesión de ese usuario.

Al guardar los datos en el archivo de sesión se actualiza su última hora de modificación, y el intérprete de PHP puede comprobar esta hora para saber si una sesión ha sido utilizada recientemente.

Sesiones

¿Cuánto duran las sesiones?

Para funcionar, una sesión necesita tanto la cookie de sesión en el navegador como el archivo de sesión en el servidor.

- Las cookies de sesión expiran cuando los usuarios cierran el navegador.
- Los archivos de sesión pueden ser borrados por el servidor si no se modifican en un periodo de tiempo (por defecto es de 24 minutos).

Sesiones

¿Cómo se inician las sesiones?

Cuando un sitio utiliza sesiones, cada página debe llamar a la función incorporada de PHP `session_start()`.

Cuando se llama a esta función, si el navegador que solicita la página no envió una cookie de sesión o si no se puede encontrar un archivo de sesión que coincida, el intérprete de PHP inicia automáticamente una nueva sesión para ese usuario.

Sesiones

Otras formas de utilizar las sesiones

En lugar de utilizar cookies de sesión, es posible añadir un identificador de sesión a las URL, pero esto es menos seguro. También es posible almacenar los datos de sesión en una base de datos (Normalmente sólo se utiliza en sitios que tienen volúmenes muy altos de tráfico y requieren varios servidores para manejar la carga).

Sesiones

Ejemplo de peticiones usando Sesiones

Primera solicitud de la página

- El navegador solicita una página utilizando una petición HTTP

Sesiones

Ejemplo de peticiones usando Sesiones

Primera solicitud de la página

En el servidor, la página PHP llama a session_start(). El navegador no envió una cookie de sesión, por lo que:

- Genera un ID de sesión para ese usuario
- Crea un archivo de sesión para guardar los datos de ese usuario (el nombre del archivo incluye el ID de sesión)

La página añade datos al superglobal `$_SESSION`. Cuando termina de ejecutarse, los valores de este array se añaden al fichero de sesión que creó para este usuario.

Sesiones

Ejemplo de peticiones usando Sesiones

Primera solicitud de la página

- El servidor devuelve la página solicitada
- Envía un HTTP que creará una **cookie de sesión** con el ID de sesión

```
RESPONSE:  page.php
HEADER:    PHPSESSID = 1234567
```

Sesiones

Ejemplo de peticiones usando Sesiones

Primera solicitud de la página

- El navegador muestra la página
- Crea una cookie de sesión con el ID de sesión

Ejemplo de peticiones usando Sesiones

Peticiones de página posteriores

- El navegador solicita una página utilizando una petición HTTP
- Envía una cabecera HTTP con el identificador de sesión

Sesiones

Ejemplo de peticiones usando Sesiones

Peticiones de página posteriores

En el servidor, la página PHP llama a `session_start()`. El intérprete de PHP encuentra el archivo de sesión con el ID de sesión especificado en la cookie de sesión y:

- Añade los datos del archivo de sesión al array superglobal `$_SESSION` para que la página pueda utilizar estos datos.
- Crea una página utilizando los datos del array
- Es capaz de actualizar los datos del array

Cuando la página ha terminado de ejecutarse, los valores del superglobal `$_SESSION` se guardan en el archivo de sesión. Esto actualiza la última hora de modificación del archivo de sesión.

Sesiones

Ejemplo de peticiones usando Sesiones

Peticiones de página posteriores

- El servidor devuelve la página solicitada

Sesiones

Ejemplo de peticiones usando Sesiones

Peticiones de página posteriores

- El navegador muestra la página
- Envía la cookie de sesión con cada solicitud al mismo sitio hasta que el usuario cierra la ventana del navegador

Cómo crear y acceder a sesiones

Cada página de un sitio que utiliza sesiones debe llamar a `session_start()`. Si el usuario no tiene sesión, inicia una por él; si la tiene, obtiene los datos de la sesión y los coloca en el array superglobal `$_SESSION`.

Cómo crear y acceder a sesiones

Cuando un visitante solicita por primera vez una página que llama a `session_start()`, se **crea un nuevo ID de sesión, una cookie de sesión y un archivo de sesión**.

```php
session_start();
```

Cómo crear y acceder a sesiones

La función debe ser llamada **antes de que cualquier contenido sea enviado al navegador** porque envía una cabecera HTTP para crear la cookie de sesión.

También debe ser llamada **antes de que la página intente obtener datos de sesión**, ya que transfiere datos del archivo de sesión al array superglobal `$_SESSION`.

Cómo crear y acceder a sesiones

Si añades datos al array superglobal `$_SESSION`, cuando la página ha terminado de ejecutarse, el intérprete de PHP añade los datos al fichero de sesión para ese usuario.

La **sintaxis para añadir datos al array es la misma que para cualquier array asociativo**. La clave debe describir los datos que el elemento está utilizando para almacenar.

El valor de cada clave puede ser un valor **escalar** (cadena, número o booleano) o un **array** u **objeto**. Este tipo de datos se conserva (a diferencia de las cookies que sólo almacenan cadenas).

```php
$_SESSION['name'] = 'Ivy';
$_SESSION['age']  = 27;
```

Cómo crear y acceder a sesiones

Cuando recojas datos del array superglobal `$_SESSION`, utiliza el operador de coalescencia nula (`??`) en caso de que falten valores o utiliza las funciones de filtro de PHP con el tipo de entrada `INPUT_SESSION`.

```php
$name = $_SESSION['username'] ?? null;
$age  = $_SESSION['age']      ?? null;
```

Cómo crear y acceder a sesiones

FUNCTION	DESCRIPTION
`session_start()`	Create new session, or get data from existing session.
`session_set_cookie_params()`	Settings used to create the session cookie (same parameters as p336).
`session_get_cookie_params()`	Returns an array holding the arguments used to set the cookie.
`session_regenerate_id()`	Creates a new session ID, and updates the session file and cookie.
`session_destroy()`	Deletes the session file from the server.

Ejemplo: Almacenando y accediendo a datos de sesiones

Este ejemplo hace el mismo trabajo que el *Ejemplo: establecer y acceder a cookies*, pero el contador se almacena en una sesión.

1. Cuando se llama a la función `session_start()` de PHP, el intérprete de PHP intenta recuperar los datos del archivo de sesión y almacenarlos en el array superglobal `$_SESSION`. Si no puede, se creará una nueva sesión para este visitante.
2. Si la clave counter del array superglobal `$_SESSION` tiene un valor, se almacena en una variable llamada `$counter`. En caso contrario, `$counter` mantiene el valor 0.

Ejemplo: Almacenando y accediendo a datos de sesiones

3. El visitante acaba de ver una página por lo que se añade 1 al contador.
4. Se actualiza el valor de la clave del contador en el array superglobal `$_SESSION`.
5. La variable `$message` almacena un mensaje indicando el número de páginas que ha visto el visitante.
6. Se muestra el mensaje.

Ejemplo: Almacenando y accediendo a datos de sesiones

Una vez que la página se ha ejecutado, PHP toma los datos del array superglobal `$_SESSION` y los guarda en el archivo de sesión para ese usuario.

Al guardar los datos también se actualiza la última hora de modificación del archivo de sesión en el servidor, por lo que los datos de la sesión durarán más tiempo.

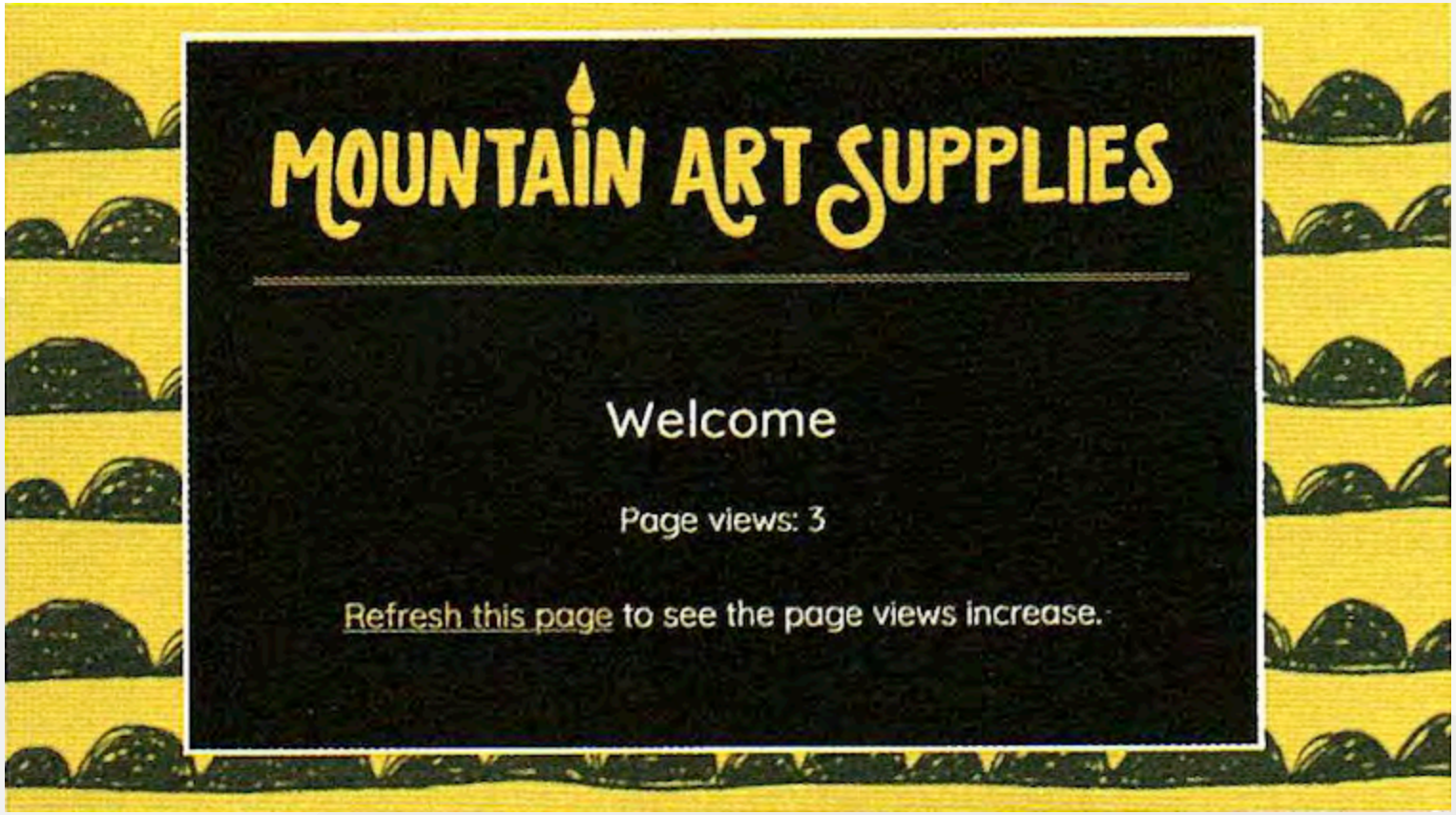
> PRUEBA: Una vez que hayas visto la página una vez, actualiza la página y observa cómo sube el contador.

> PRUEBA: Almacena tu nombre en el array superglobal `$_SESSION`, y muéstralo en la página.

Ejemplo: Almacenando y accediendo a datos de sesiones

```
<?php
① session_start();                          // Create/resume
② $counter = $_SESSION['counter'] ?? 0;     // Get data
③ $counter = $counter + 1;                  // Counter + 1
④ $_SESSION['counter'] = $counter;          // Update session

⑤ $message  = 'Page views: ' . $counter;    // Message
?>
<?php include 'includes/header.php'; ?>

<h1>Welcome</h1>
⑥ <p><?= $message ?></p>
<p><a href="sessions.php">Refresh this page</a> to see
the page views increase.</p>

<?php include 'includes/footer.php'; ?>
```

Ejemplo: Almacenando y accediendo a datos de sesiones



Actividad: Carrito de compras simple

Imaginemos que estamos trabajando en una tienda en línea donde los usuarios pueden añadir productos a su carrito de compras antes de proceder al pago. El objetivo es que los productos seleccionados permanezcan en el carrito mientras el usuario navega por el sitio, utilizando sesiones para almacenar los datos temporalmente.

Actividad: Carrito de compras simple

Instrucciones

1. **Crear la Página de Productos (products.php):**
    - Crear una lista de productos ficticios, cada uno con un nombre, precio, y un botón para añadirlo al carrito.
    - Cuando el usuario haga clic en "Añadir al carrito," se añadirá el producto a la sesión y se redirigirá a una página que muestra los productos añadidos.

Actividad: Carrito de compras simple

2. **Configurar la Página del Carrito (cart.php):**
    - La página del carrito mostrará los productos seleccionados por el usuario, almacenados en la sesión.
    - Incluir un contador para cada producto que muestre la cantidad seleccionada y el precio total acumulado de cada uno.

Actividad: Carrito de compras simple

3. **Finalizar la Compra (checkout.php):**
    - Crear una opción para "vaciar el carrito" o proceder a la compra, que eliminará todos los datos de la sesión y confirmará la acción al usuario.

La vida de una sesión

Los navegadores eliminan las cookies de sesión cuando se cierra la ventana del navegador. Los servidores eliminan los archivos de sesión cuando se ejecuta el proceso de **recogida de basura** (**garbage collection**). Como resultado, las sesiones pueden durar más de lo esperado.

La vida de una sesión

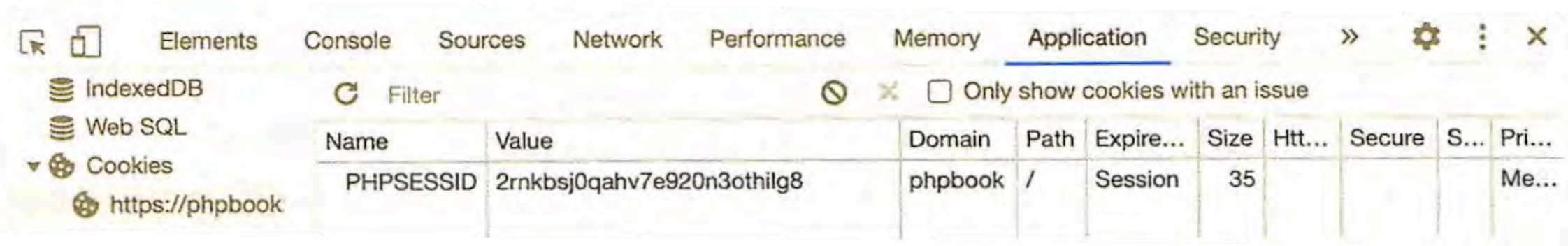
Si aún no lo has hecho, abre el ejemplo anterior en tu navegador. A continuación, abre:

- Las herramientas de desarrollo del navegador para que muestren las cookies
- La carpeta donde el servidor web almacena los archivos de sesión

Si necesitas ayuda para encontrar los archivos de sesión, consulta el *Anexo I: Cómo encontrar los archivos de sesión*.

La vida de una sesión

En el navegador, deberías ver una cookie llamada `PHPSESSID`. El valor de esa cookie es el **ID de sesión**.



La vida de una sesión

En la carpeta donde el servidor web almacena los archivos de sesión debería ver un nombre de archivo que contiene el ID de sesión. Anota la fecha y hora en que se modificó por última vez el archivo de sesión y, a continuación, actualiza el navegador que muestra el ejemplo anterior; **la hora de la última modificación se actualizará**.

Si abres el ejemplo en un navegador web diferente (por ejemplo, prueba Chrome y Firefox), **se creará una nueva sesión porque la sesión está vinculada al navegador**.

La vida de una sesión

Cuando una página llama a `session_start()`, si el intérprete de PHP no recibe una cookie de sesión o no puede encontrar un archivo de sesión que coincida con esa cookie de sesión, **creará una nueva sesión**.

Cuando una página que ha llamado a session_start() termina de ejecutarse, guarda los datos del array superglobal `$_SESSION` en el fichero de sesión. **Esto actualiza la última hora de modificación del archivo de sesión**.

La vida de una sesión

El intérprete de PHP utiliza la fecha y hora en que la sesión fue modificada por última vez para **determinar cuándo puede borrar el archivo de sesión** (lo que finalizaría la sesión).

Por lo tanto, **cuando un sitio utiliza sesiones, es importante llamar a `session_start()` en cada página**.

De lo contrario, si el usuario estaba navegando por páginas del sitio que no actualizaron esta configuración, **la sesión podría finalizar mientras el usuario aún está navegando por el sitio**.

La vida de una sesión

El servidor web ejecuta un proceso llamado **recolección de basura**. Elimina los archivos de sesión cuya última fecha de modificación supere un tiempo determinado (**por defecto, 24 minutos**).

Una vez **borrado un archivo de sesión, la sesión finaliza** porque, aunque un navegador enviara la cookie de sesión, el archivo que contiene los datos de sesión no se encontraría.

La vida de una sesión

Comprobar la última vez que se accedió a cada archivo de sesión y borrar los archivos de sesión antiguos consume recursos del servidor, por lo que los servidores **intentan no hacerlo con demasiada frecuencia**.

La frecuencia con la que se ejecuta la recolección de basura depende del número de veces que se accede a las sesiones.

La vida de una sesión

PHP no ejecuta el proceso de recolección de basura en cada solicitud de sesión de forma estricta. En lugar de eso, PHP **aplica una probabilidad**, o "probabilidad de ejecución", **para decidir cuándo limpiar las sesiones inactivas**.

La vida de una sesión

Esta probabilidad se define en la configuración de PHP mediante dos parámetros:

1. **session.gc_probability**: Define la probabilidad de que se ejecute el recolector de basura en cada solicitud de sesión.
2. **session.gc_divisor**: Junto con `gc_probability`, controla esa probabilidad en forma de fracción.

La vida de una sesión

Por ejemplo, si `session.gc_probability` es 1 y `session.gc_divisor` es 100, entonces cada vez que el servidor recibe una solicitud que involucra sesiones, el recolector de basura se ejecutará con una probabilidad de 1/100, es decir, un 1% de las veces.

La vida de una sesión

Cómo influye la actividad del sitio

- En sitios **con mucho tráfico**, el recolector de basura se activará más frecuentemente debido al mayor número de solicitudes que involucran sesiones, lo cual incrementa las probabilidades de limpieza regular.
- En sitios **con poco tráfico**, al haber menos solicitudes que activan sesiones, el recolector de basura se ejecutará menos frecuentemente. Esto puede resultar en archivos de sesión que permanecen más tiempo en el servidor, ya que la limpieza no se dispara con frecuencia.

La vida de una sesión

Este sistema permite reducir la carga del servidor, ya que la limpieza de archivos de sesión solo se hace cuando es necesario, sin que el servidor tenga que realizar el proceso en cada solicitud de sesión.

Por lo tanto, **en un sitio tranquilo, la recolección de basura puede no ejecutarse durante horas o incluso días**.

La vida de una sesión

Como se ve en el siguiente ejemplo, las sesiones se utilizan a menudo para recordar cuándo un usuario ha iniciado sesión en un sitio. En estos casos, los usuarios deben tener la opción de cerrar la sesión.

Si un usuario no cierra la ventana de su navegador (por lo que el navegador sigue enviando una cookie de sesión) **y el servidor está quieto** (por lo que no ha ejecutado la recolección de basura) **la sesión podría durar más de lo que fue diseñada**.

Esto es un gran **problema cuando los usuarios comparten ordenadores** porque, si un usuario no cierra la sesión, **otra persona podría visitar el sitio utilizando su cuenta**.

La vida de una sesión

Terminar una sesión implica los cuatro pasos siguientes:

1. **Eliminar todos los datos del archivo de sesión estableciendo el array superglobal `$_SESSION` a un array en blanco**. Esto también impide que cualquier código posterior en la misma página acceda a esos valores.

```
$_SESSION = [];
```

La vida de una sesión

2. En el paso 3, se utilizará `setcookie()` para actualizar la cookie de sesión. Cuando se utiliza, los argumentos para los parámetros *path*, *domain*, *secure* y *httponly* deben utilizar los mismos valores que se utilizaron cuando se creó la cookie. La función `session_get_cookie_params()` de PHP **devolverá los valores que se utilizaron para estos parámetros cuando se creó la cookie de sesión**. Los valores que devuelve se almacenan como un array en la variable `$params`.

```php
$params = session_get_cookie_params();
```

La vida de una sesión

3. La función `setcookie()` de PHP se utiliza para actualizar la cookie de sesión. El **parámetro `value` se establece en una cadena en blanco**; esto borra el ID de sesión de la cookie de sesión.

   El parámetro `expires` se establece en una **fecha en el pasado para evitar que el navegador envíe la cookie al servidor si solicita más páginas**. Todos los **demás parámetros se establecen utilizando los valores recogidos en el paso 2** y almacenados como un array en la variable `$params` .

```
setcookie('PHPSESSID', '', time() - 3600, $params['path'],
    $params['domain'], $params['secure'], $params['httponly']);
```

La vida de una sesión

4. Llama a la función `session_destroy()` de PHP para i**ndicar al intérprete PHP que elimine el archivo de sesión**. El intérprete de PHP **borrará el archivo inmediatamente en lugar de esperar a la recolección de basura** para borrarlo.

```
session_destroy();
```

Anexo I: Cómo encontrar los archivos de sesión

Si estás desarrollando un sitio que utiliza sesiones, es posible que desees comprobar que los archivos de sesión se están creando correctamente, y luego comprobar su última hora de modificación para asegurarte de que se están actualizando.

Anexo I: Cómo encontrar los archivos de sesión

Para ello, necesitas saber dónde guarda tu servidor web los archivos de sesión. Si estás ejecutando MAMP o XAMPP localmente, las rutas por defecto para los archivos de sesión son:

- MAMP en un Mac: `/Aplicaciones/MAMP/tmp/php/`
- XAMPP en un PC: `C:\xampp\tmp`

Anexo I: Cómo encontrar los archivos de sesión

El archivo `php.ini` especifica la ubicación donde el servidor web guarda los archivos de sesión; se almacena en un parámetro llamado `session.save_path`

Para comprobar el valor de este ajuste, puedes comprobarlo en el archivo `php.ini` o llamar a la función `phpinfo()` de PHP, y luego buscar el término `session.save_path`.

Anexo I: Cómo encontrar los archivos de sesión

También puedes utilizar la función `ini_get()` de PHP para encontrar la configuración actual (como se muestra a continuación):

```php
<?php
  echo ini_get('session.save_path');
?>
```

> NOTA: No debes dejar un archivo que exponga la ubicación de los archivos de sesión en un servidor público.