La Sénia
Institut d'Educació Secundària
Paiporta

UP3-3. Mantenimiento del estado: Cookies y Sesiones

Desarrollo Web en Entorno Servidor

Profesora: Silvia Vilar Pérez

curso 2025-2026

Contenidos

- Mantenimiento del estado.
- Cookies.
- Sesiones.

Mantenimiento del estado

- El mantenimiento del estado de la conexión entre cliente y servidor puede hacerse mediante dos formas: ***Cookies*** o ***Sesiones***. La primera, se almacenarán los datos en el cliente y en la segunda, los datos del estado se guardarán en el servidor
- Por motivos de persistencia y seguridad, es aconsejable usar sesiones ya que el cliente puede rechazar las cookies o bien pueden ser borradas.
- La sesión es como se denomina a la conexión entre cliente y servidor. Cada sesión puede abarcar múltiples páginas web y su seguimiento se realiza mediante la gestión del estado de la conexión y los datos.

Mantenimiento del estado

Muchas veces es necesario mantener el estado de una conexión entre distintas páginas o entre distintas visitas a un mismo sitio, de modo que se conserven los datos generados.

Algunas situaciones en las que se requiere mantener el estado son, por ejemplo, con la finalidad de:

- Acumular una cesta de la compra
- Control de acceso de los usuarios
- Conocer los pasos de la navegación y preferencias del usuario
- Actualizar una base de datos
- Etc.

Cookies

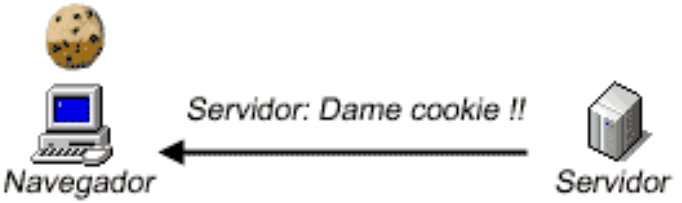
El manejo de las Cookies en PHP es sencillo:

- Primero se envía la Cookie, invocando a la función **setcookie()** para crear la Cookie en el cliente
- En las posteriores peticiones que recibamos de ese cliente vendrá incrustada la Cookie. Se consulta la información almacenada en ella accediendo a la superglobal **$_COOKIE**.
- Las Cookies se envían en las cabeceras de las transacciones HTTP y deben enviarse antes que cualquier otra cabecera HTML (restricción de las Cookies no de PHP)

Cookies - Proceso

1) La Cookie es enviada al navegador desde el servidor y, si la acepta, permanece en el cliente

2) Las páginas ejecutadas en el servidor piden la Cookie al navegador



3) El navegador envía la Cookie permitiendo la identificación del usuario por parte del servidor

Cookies - Campos

Las Cookies son bloques de texto sin formato, de tamaño máximo 4KB y formadas por varios campos:

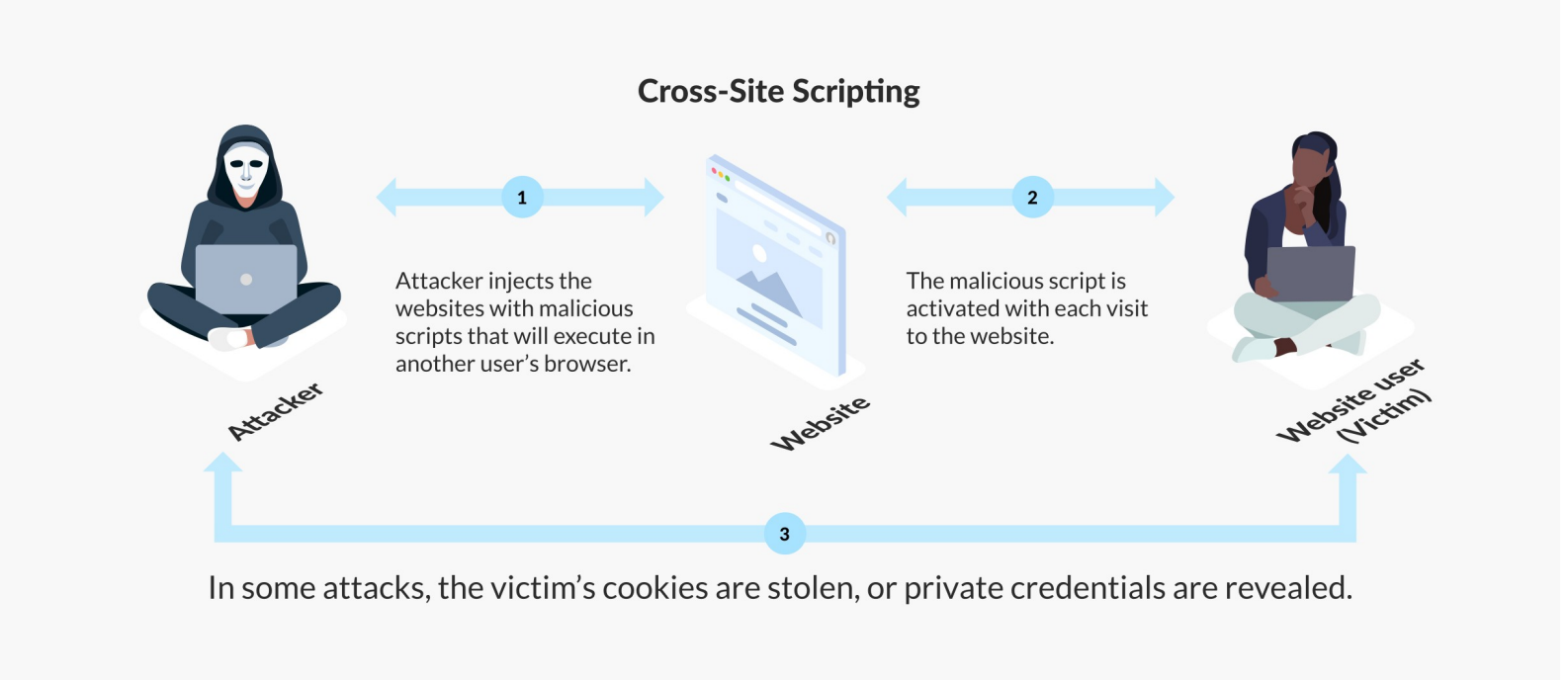
- **Nombre** (requerido): Nombre de la Cookie
- **Valor**: Valor asociado con codificación URL
- **Fecha Expiración**: Momento en el que deja de ser válida
- **Path**: Subconjunto de URLs para los que la Cookie es válida
- **Dominio**: rango de dominios para los que la cookie es válida en el servidor
- **Segura**: Indica si la cookie se debe transmitir exclusivamente sobre https
- **Httponly**: Accesible sólo por el protocolo HTTP (no accesible desde lenguajes script como JavaScript)

Ver RFC6265 http://www.faqs.org/rfcs/rfc6265.html

Cookies - Campos

Al indicar **Httponly** en la cookie, sólo se puede acceder a la misma usando el protocolo HTTP, es decir, desde el cliente no se puede acceder mediante JavaScript evitando así ataques ***XSS (Cross-Site Scripting)*** y protegiendo información crítica como un ID de sesión, un token de autenticación, datos del usuario, etc.

Cross-Site Scripting: inyección de código malicioso en el navegador o, en este caso, en la cookie



Cookies - Salvedades

- Las Cookies ***no son visibles hasta la próxima carga*** de la página en la que debieran serlo. Para probar si se ha creado, se debe buscar la cookie en alguna página cargada posteriormente y antes que la cookie expire.

  Podemos comprobar las Cookies desde PHP con:
  echo $_COOKIE["MyCookie"];** o **print_r($_COOKIE);
- Al ***borrar*** una Cookie hay que ***asegurar que ha expirado***, por ejemplo asignando $_COOKIE expires a un momento previo.
- Si a $_COOKIE value se asigna FALSE (o **‘’**) y los demás campos tienen los mismos valores que en la llamada anterior, intentará eliminar la cookie. Por ello, al configurar valor, no se debe usar valores booleanos sino indicar 0 para FALSE y 1 para TRUE.
- Múltiples llamadas a setcookie() se efectúan en el orden de llamada.

Cookies - Ejemplo

```
<?php
$cookie_name = "user";
$cookie_value = "Silvia";
$cookie_expires=time() + (60*60*24 * 30) //30 días
$cookie_path= "/" //Todo el sitio Web
setcookie($cookie_name, $cookie_value,$cookie_expires,$cookie_path);
// la cookie se debe crear previamente a cualquier cabecera o html
?>
<html>
<body>
<?php
if(!isset($_COOKIE[$cookie_name])) {
  echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
  echo "Cookie '" . $cookie_name . "' is set!<br>";
  echo "Value is: " . $_COOKIE[$cookie_name];
}
?>
<p><strong>Note:</strong> You might have to reload the page to see the value of the cookie.</p>
</body>
</html>
```

Cookie 'user' is set!
Value is: Silvia
Note: You might have to reload the page to see the value of the cookie.

Cookies - Borrado

Podemos diferenciar las siguientes situaciones:

1) Queremos **borrar los parámetros** de la Cookie: Se invoca a setcookie() sin parámetros

```
<?php
    setcookie("user")
?>
```

2) Queremos **eliminar del servidor la variable cookie** ya leída: Usamos la función unset()

```
<?php
    unset($_COOKIE["user"])
?>
```

3) Queremos **eliminar el archivo Cookie del cliente**: Indicaremos un tiempo de expiración 0 o anterior al actual

```
<?php
    setcookie("user","Silvia",0) // o time()-Valor_en_segundos
?>
```

Cookies - Ejemplo

- Para borrar la cookie invocamos setcookie con los parámetros value=”” y $expires a un tiempo anterior al presente:

```
<?php
$cookie_name = "user";
$cookie_value = """; //también funciona con FALSE
$cookie_expires=time() -3600 //Hace una hora
$cookie_path= "/" //Todo el sitio Web
setcookie($cookie_name, $cookie_value,$cookie_expires,
$cookie_path);
//También podríamos invocar:
setcookie($cookie_name,””,time() -3600);
?>
```

- Si ***no se indica fecha de expiración*** o el ***valor indicado es 0***, la cookie expirará (se borra) tras cerrar la sesión (cierre del navegador)

Sesiones

- Las sesiones son una manera de guardar información específica para cada usuario durante toda su visita.
- Cada usuario que entra en un sitio abre una sesión, independiente de la sesión de otros usuarios.
- En la sesión de un usuario podemos almacenar todo tipo de datos: nombre del usuario, productos de un carrito de la compra, preferencias de visualización o trabajo, páginas por las que ha navegado, etc.
- Todas estas informaciones se guardan en lo que denominamos variables de sesión.

Sesiones vs Cookies

- Las sesiones mantienen el valor de las variables a lo largo de toda la navegación.
- Las cookies permiten almacenar poca información (sólo 4KB), las sesiones no tienen esa limitación.
- Muchos usuarios/ navegadores desactivan las Cookies.
- Las sesiones almacenan la información en el servidor en lugar de almacenarla en el cliente como las Cookies. Su ubicación se indica en ***session.save_path*** en php.ini
- PHP internamente genera un identificador de sesión único **(SID, Session ID)**, que sirve para determinar las variables de sesión que pertenecen a cada usuario. Se accede desde la cookie **PHPSESSID**
- Para propagar el SID al cliente y que lo indique en su visita, bien se guarda en una Cookie o bien se pasa por parámetro en la URL (query string).

Sesiones vs Cookies

Las sesiones funcionan aunque las Cookies estén desactivadas. Si el servidor web detecta que las cookies no están activadas, añade ***automáticamente*** el id de sesión como un **query string** ***nombre_sesión=SID*** en todos los enlaces de la página. Para que esto funcione hay que cumplir estos requisitos:

- Todas las páginas tienen que tener la extensión **.php**, para que php pueda añadir la SID a las páginas.
- Es necesario que ***session.use_trans_sid*** esté activado y ***session.use_only_cookies*** esté desactivado en **php.ini.**
  https://www.php.net/manual/es/session.configuration.php#ini.session.use-trans-sid
- Se pueden generar en anchors (etiquetas <a>) y formularios pasando automáticamente la constante SID, pero no así con header(location) que debe propagase de forma manual.
- Sólo se aplica en **direcciones relativas** del servidor, no absolutas (servidores externos) por seguridad.
- Usar **htmlspecialchars(SID)** para prevenir ataques

Sesiones

- Los datos entre peticiones de la sesión se almacena en la variable superglobal **$_SESSION** y permanecen hasta que se cierra el navegador.
- Cuando un cliente accede a un sitio web PHP comprobará, automáticamente (si **session.auto_start** está establecido a 1) o sobre su petición (explícitamente a través de **session_start()**), si se ha enviado un id de sesión específico con la petición. Si éste es el caso, se recrea el entorno anteriormente guardado.

Ver aspectos de seguridad en sesiones:
https://www.php.net/manual/es/session.security.php

Sesiones: Funciones

session_start() crea una sesión o reanuda la actual basada en el SID pasado mediante una petición GET/POST o bien una cookie.

session_name() establece o devuelve el valor nombre de la sesión. Por defecto es PHPSESSID.

session_id() Devuelve y/o establece el identificador de la sesión.

Sesiones – Pasando el SID

PHP añade ***automáticamente*** el SID de forma transparente al programador según la configuración de ***session.trans_sid_tags*** (en las etiquetas HTML a=href, area=href, frame=src, input=src, form=). <input hidden="id_sesión" name="nombre_sesión"> se añade como variable de formulario .Por seguridad, los formularios deben enviarse con método POST de modo que el SID no aparezca en la URL.

Ejemplo manual del efecto de tener esta opción activada:

- En nuestro archivo index.php indicamos el SID:

  <**a href**="otrapagina.php?**<?php print SID; ?>**">Otra página

  <a href="otrapagina.php?**<?php print session_name(); ?>**">Otra página // nos referimos a la sesión con un nombre

  <a href="otrapagina.php?**<?php print session_id(); ?>**">Otra página //Otro modo de recuperar el SID

- Al navegador le llegará de la siguiente forma:

Otra página

Sesiones – Pasando el SID

```
<?php
ini_set(“session.use_trans_sid”, “1”); //activamos sesión si fallan cookies
ini_set(“session.use_cookies”, “0”); //desactivamos el uso de cookies
ini_set(“session.use_only_cookies”, “0”); //desactivamos obligación de usar sólo cookies
session_start(); //iniciamos la sesión
$_SESSION['dato'] = “cliente nuevo”; //ponemos un dato de prueba
?>
<html>
<form action="nextpage.php?<?php echo htmlspecialchars(SID); ?>" method="POST">
    Introduce el dato: <input type="Text" name="dato">
</form>
</html>
```

Sesiones – Pasando el SID

En nextpage.php:

```
<?php
session_start(); //iniciamos la sesión
$_SESSION['dato'] = $_POST['dato'];

if (empty($_SESSION['dato'])) {
   echo "No hay dato introducido";
} else {
   echo "El dato introducido es: ", $_SESSION['dato'];
}
// Cambia "cliente nuevo" por el valor introducido en el Text
?>
```

En la url de la página nextpage.php a la que hemos enviado el dato nuevo, podéis observar el valor de PHPSESSID

El dato introducido es: Dato nuevo

Sesiones – Ejemplo Sesión

```
<?php
ini_set("session.use_trans_sid", "1"); //activamos sesión si fallan cookies
ini_set("session.use_cookies", "0"); //desactivamos el uso de cookies
ini_set("session.use_only_cookies", "0"); //No usar sólo cookies
session_start(); //iniciamos la sesión
if (empty($_SESSION['count'])) {    //contaremos las veces que visita la página
    $_SESSION['count'] = 1;
} else {
    $_SESSION['count']++;
}
?>
<html>
<body>
    <p>
        Hola visitante, ha visto esta página <?php echo $_SESSION['count']; ?>
veces.
    </p>
    <p>
        Para continuar, <a href="nextpage.php?<?php echo htmlspecialchars(SID);
?>">haga clic aquí</a>.
    </p>
</body>
</html>
```

Sesiones – Funciones de uso

Funciones de PHP para el manejo de sesiones:

- Registra o modifica una variable de sesión

$_SESSION['nombre'] = valor;

- Elimina una o todas las variables de sesión

unset ($_SESSION['nombre']);

Otra forma menos elegante, asignar vector vacío:

$_SESSION=array();

- Comprueba si la variable de sesión está creada

if (isset($_SESSION['nombre']))

Sesiones – Funciones de uso

Funciones de PHP para el manejo de sesiones:

- **session_cache_expire()**: Devuelve/asigna la caducidad de la caché actual.
- **session_cache_limiter()**: Obtener y/o establecer el limitador de caché actual.
- **session_commit()**: alias de session_write_close. Guarda los datos de la sesión y la cierra
- **session_decode()**: decodifica la información de sesión desde una cadena.
- **session_encode()**: codifica la información de la sesión actual y los devuelve como una cadena.
- **session_regenerate_id()**: Actualiza el id de sesión actual con un nuevo valor más reciente.
- **session_create_id()**: Genera un nuevo SID.

Sesiones – Funciones de uso

- **session_save_path()**: Obtiene y/o establece la ruta de almacenamiento de la sesión actual.
- **session_write_close()**: Escribe la información de sesión y finalizar la sesión.
- **session_reset()**: Reinicializa el array de sesión con los valores originales
- **session_unset()**: Libera las variables de la sesión
- **session_abort()**: Desecha los cambios en el array de sesión y la finaliza
- **session_destroy()**: Destruye toda la información asociada con la sesión actual pero no destruye las variables globales asociadas con la sesión, ni destruye la cookie de sesión. Para volver a utilizar las variables de sesión se debe llamar a session_start(). Para eliminar por completo la cookie con el SID en el cleinte se debe usar setcookie()

Sesiones – Funciones de uso

- **session_get_cookie_params()**: Devuelve una matriz asociativa con la información de la cookie de la sesión. Se puede usar para obtener los parámetros y así pasarlos para caducar la sesión.

Alguna información que contiene es:

  - 'lifetime': Tiempo de vida de la cookie de sesión
  - 'path': Ruta donde se guarda la cookie de sesión.
  - 'domain': Dominio del servidor que genera la cookie.
  - 'secure': Sólo se puede enviar en conexiones seguras
  - 'httponly': La cookie sólo se propaga por HTTP (no acceso desde lenguajes script como JS).

- **session_set_cookie_params()**: Se establecen los parámetros de la sesión antes de llamar a session_start(), Este efecto sólo dura para cada petición durante la ejecución del script

Sesiones – Funciones de uso

/* Ejemplo de uso para **eliminar la cookie de sesión** con los mismos parámetros de creación (recordad que session_destroy no elimina la cookie de sesión ni sus variables globales asociadas) */

```
// Obtenemos los datos de la sesión actual
$CookieInfo = session_get_cookie_params();

if ( (empty($CookieInfo['domain'])) && (empty($CookieInfo['secure'])) ) {
      //Si no hay valores específicos para domain y secure, no se indican
      setcookie(session_name(), '', time()-3600, $CookieInfo['path']);
} elseif (empty($CookieInfo['secure']))
      { //No se indican valores para secure
            setcookie(session_name(), '', time()-3600,$CookieInfo['path'],
                  $CookieInfo['domain']);
      } else { //se indican todos los valores que tenía la cookie de sesión
                  setcookie(session_name(), '', time()-3600, $CookieInfo['path'],
                        $CookieInfo['domain'], $CookieInfo['secure']);
                  }
session_destroy();
```