Desarrollo web en entorno servidor

UT 1.1 – Arquitectura y herramientas web
7 – PHP – Cookies – Sesiones

IES Clara del Rey – Madrid

HTTP es un protocolo sin estado

HTTP está diseñado como un protocolo sin estado.

Esto significa que las peticiones son independientes entre sí. Una petición es independiente de las anteriores y de las siguientes.

En este contexto, ¿cómo se pueden relacionar todas las peticiones de un usuario para mantener un estado? Por ejemplo, para:

- Identificar al usuario conectado y mantener su sesión.
- Mantener elementos como carritos de compra.
- Mantener las preferencias de usuario, configuraciones, etc.

Las cookies se utilizan para dar "memoria" a HTTP.

Cookies

Una cookie es un pequeño archivo de texto que el servidor almacena en el navegador cuando se visita un sitio web.

El servidor indica cómo almacenar la cookie con la cabecera "set-cookie".

Un mismo servidor puede almacenar varias cookies en el navegador del cliente. Por ejemplo, para mantener sesión, para guardar opciones de personalización, para registrar el rechazo de cookies de seguimiento, etc.

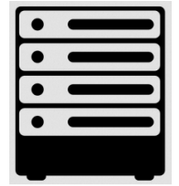
El navegador, cada vez que pide una página del mismo sitio web, envía la cookie (o cookies) al servidor, junto al resto de la petición.

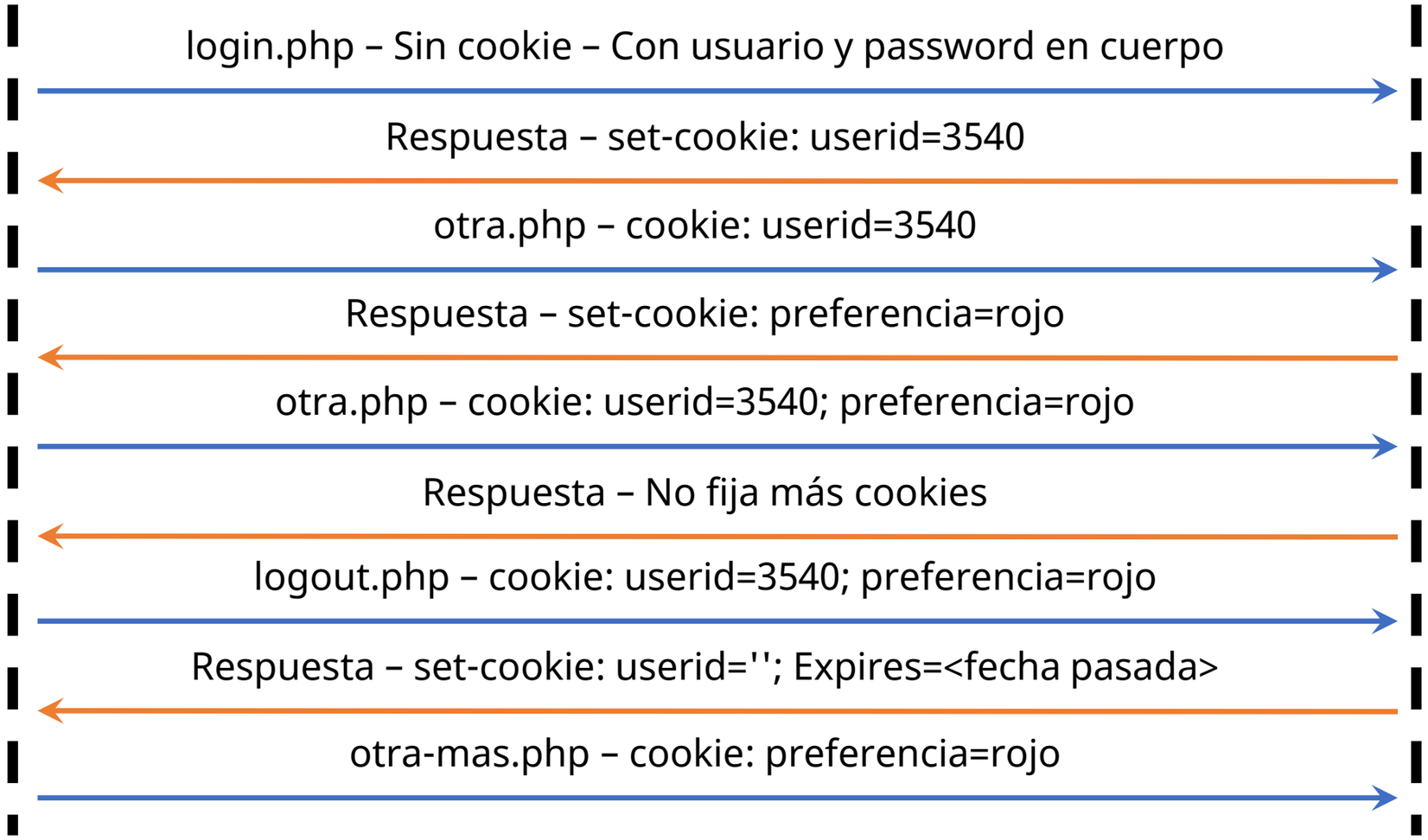
Para el envío de las cookies, se usa una cabecera "cookie".

Cookies – Ejemplo de funcionamiento

Cliente
(navegador)

Servidor
Web – HTTP





Cookies – Ejemplo de funcionamiento

En el diagrama de ejemplo anterior:

1. El cliente envía una petición a la página "login.php", con los datos de usuario y contraseña. Esta petición no lleva cookies.
2. El servidor valida los datos. Si son correctos en la respuesta incluye una cabecera set-cookie, con:
   Nombre de la cookie: userid
   Valor de la cookie: 2540
   Esta cookie sirve para mantener la sesión del usuario entre peticiones. A partir de este momento, todas las peticiones que el cliente realice llevarán la cookie "userid".
3. El cliente pide una página "otra.php". Como el servidor había fijado la cookie "userid", el cliente la envía al servidor.

Cookies – Ejemplo de funcionamiento

4. El servidor responde con otra cookie más (preferencia=rojo), que se añade a la ya existente.
5. El cliente hace más peticiones, y en todas ellas envía las cookies.
6. El cliente accede a logout.php, que sirve para cerrar la sesión.
7. El servidor responde con una cabecera set-cookie, con:
   Nombre de la cookie: userid
   Valor de la cookie: vacío, aunque no es imprescindible hacerlo.
   Expires: una fecha del pasado. Para que caduque la cookie.
8. El navegador cumple con lo que dice la última cabecera para la cookie userid, y como ya ha expirado, no la enviará en posteriores peticiones.

Cookies – Propiedades

Como se puede ver, además del valor de la cookie, se pueden añadir propiedades como la fecha en la que expira. Algunas propiedades:

- Domain – Dominio de la cookie. Limita a que servidores se envía.
- Path – Limita a que partes del sitio web se envía la cookie.
- Expires – Define cuándo caduca la cookie.
- HttpOnly – Impide que JavaScript acceda a la cookie.
- SameSite – Determina si la cookie puede mandarse en peticiones a otros dominios. Relacionada con la protección contras XSS.
- Secure – La cookie se enviará solo si se usa HTTPS.

Más información en MDN: Información de set-cookie en MDN

PHP – Cookies – Leer cookies recibidas

Para leer cookies recibidas se utiliza el array superglobal $_COOKIE.

Es un array asociativo en el que se almacenan todas las cookies que llegan en la petición, y a las que se puede acceder por el nombre de la cookie. Ejemplo:

```
<?php
if(isset($_COOKIE['usuario'])) {
        $nombreUsuario = $_COOKIE['usuario'];
        echo "Bienvenido, " . $nombreUsuario;
}
?>
```

Como en $_GET y $_POST, hay que verificar que la cookie existe con isset. Si no se hace, se puede producir un warning.

PHP – Cookies – Fijar cookies en respuesta

Las cookies se fijan en el servidor, para que se envíen al cliente, y este las almacene el tiempo necesario, según lo que haya establecido el servidor.

Para fijar una cookie se usa la función setCookie. Sintaxis:

```
bool setcookie(string $name [, string $value = ""
        [, int $expire = 0 [, string $path = "" [, string $domain = ""
        [, bool $secure = false [, bool $httponly = false ]]]]]])
```

Como se puede ver, el único parámetro obligatorio es el nombre de la cookie, y a partir de ahí, se pueden ir añadiendo parámetros para añadir atributos a la cookie.

PHP – Cookies – Fijar cookies en respuesta

Parámetros de setCookie.

- $name – Nombre de la cookie.
- $value – Valor que se almacena en la cookie.
- $expire – Tiempo de caducidad de la cookie en formato unix timestamp (segundos desde epoch – cero horas del 1/1/1970).
- $path – Ruta en la que la cookie estará disponible. Por defecto, la ruta actual. Si queremos que esté disponible en todo el sitio, se usa "/".
- $domain – Dominio para el que la cookie es válida.
- $secure – Si la cookie sólo puede enviarse por HTTPS.
- $httponly – Si es true, la cookie no podrá consultarse con JavaScript.

PHP – Cookies – Fijar cookies en respuesta

Algunas cosas a tener en cuenta:

- Siempre hay que llamar a setCookie antes de escribir HTML. Esta función fija cabeceras, y si se escribe algo de HTML antes de fijar las cabeceras se puede producir un error.
- Al llamar a setCookie, si ya existía una cookie con ese nombre, se modifica el valor de la cookie.
- Para el parámetro $expire, se puede usar time() + el tiempo de vida en segundos de la cookie. Ej: time() + 3600 es una hora de vida.
- Para eliminar una cookie, se puede usar time() – 3600, por ejemplo.
- En las cookies hay que evitar datos sensibles, porque se pueden leer fácilmente. Si hay que guardar un dato sensible se podría encriptar, pero no está recomendado.

PHP – Cookies – Cookies de sesión

Si al fijar una cookie no se especifica $expire o se usa cero (que es el valor por defecto), se crea una cookie de sesión.

Las cookies de sesión no tienen una fecha de caducidad específica.

La vida de una cookie de sesión es la duración de la sesión del navegador.

La cookie se mantendrá mientras el navegador siga abierto.

Cuando se cierra la ventana del navegador, se elimina la cookie de sesión.

Estas cookies pueden ser más seguras, porque el navegador no las guarda en el disco del cliente, las elimina al cerrar.

Son las cookies ideales para la gestión de las sesiones de usuario.

Sesiones

En una aplicación web, la sesión permite mantener el estado y los datos de un usuario a lo largo de múltiples solicitudes y respuestas al servidor.

Las sesiones se pueden mantener con varios mecanismos, pero el más habitual es usar cookies.

El funcionamiento es:

- Cuando el usuario accede, se genera un identificador de sesión. Un ID único para el usuario, que se envía al navegador en una cookie.
- Todas las peticiones posteriores incluyen el identificador de sesión, lo que permite "relacionar" unas peticiones con otras.
- Para cerrar la sesión, se puede eliminar la cookie al cerrar el navegador, o el servidor puede forzar la eliminación de la cookie

PHP – Sesiones

PHP ofrece una serie de mecanismos para gestionar la sesión sin necesidad de realizar operaciones directamente con las cookies.

Para iniciar la sesión, se debe llamar a "session_start()". Importante:

- Hay que llamar a esta función en cualquier script (fichero php) en el que se utilicen sesiones. Ya sea iniciarlas o acceder a datos de una.
- Como con otras funciones que gestionan cabeceras, hay que llamar a la función antes de generar HTML. Lo más habitual es llamarla en la primera línea del fichero PHP.

PHP – Sesiones

Se usa el array asociativo $_SESSION para almacenar y recuperar datos de la sesión.

Guardar datos en la sesión:

```
$_SESSION['usuario'] = 'Juan';
$_SESSION['email'] = 'juan@example.com';
```

Leer datos de la sesión:

```
if(isset($_SESSION['usuario'])) {
    echo "Bienvenido, " . $_SESSION['usuario'];
} else {
    echo "No has iniciado sesión.";
}
```

PHP – Sesiones

Para eliminar todas las variables de sesión, se puede vaciar el array de variables de sesión:

```
$_SESSION = array();
```

Para cerrar la sesión, se llama a la función session_destroy();

Las sesiones se pueden configurar de dos formas:

- Con el fichero php.ini, que veremos más adelante.
- Con la función ini_set($parametro, valor), que permite sobrescribir los valores que se especifican en el fichero php.ini
- Con algunas otras funciones, como session_name($name), o session_set_cookie_params($parametros)

PHP – Sesiones – Buenas prácticas

Algunas buenas prácticas cuando se usan sesiones en PHP:

- Configurar la cookie de sesión para que sea lo más segura posible:

```
# No JavaScript, solo HTTPS, no envío a terceros
session_set_cookie_params(
        ['httponly' => true, 'secure' => true, 'samesite' => 'Strict']);
session_start();
```

- Regenerar el id de sesión tras eventos "críticos" (log-in / log-out):

```
session_start();
// Proceso de autenticar al usuario
session_regenerate_id(true);
```