Prácticas

Estilo: Normal ▾ **Tamaño letra:** Normal ▾ **Interlineado:** 1.2 ▾

Desarrollo de Aplicaciones Web

Práctica 8: PHP 2 (cookies y sesiones)

1. Objetivos

- Conocer el concepto de *cookie* y sus posibles usos.
- Aprender a utilizar las *cookies* con PHP.
- Conocer el concepto de sesión en una aplicación web y sus posibles usos.
- Aprender a utilizar las sesiones con PHP.

2. Recursos

¿Qué son las cookies?

- **RFC 2965 HTTP State Management Mechanism** (1) : documento oficial que especifica el uso de las cookies para lograr una comunicación con HTTP que conserve el estado.
- **Cookie** (2) : definición en la Wikipedia de cookie, explica qué son, cómo se usan, algunos inconvenientes y alternativas a su uso.
- **Cookie Central** (3) : explica qué son las cookies y contiene ejemplos de uso.

¿Cómo se emplean las cookies con PHP?

- **PHP Cookies** (4) : documentación oficial del uso de las cookies en PHP.
- **PHP Cookies** (5) : cómo utilizar las cookies en PHP.
- **PHP y cookies** (6) : explicación en español del uso de las cookies en PHP.

¿Qué son las sesiones?

- **Aspectos básicos de las variables de sesión** (7) : explica los conceptos básicos sobre las variables de sesión y su uso en las aplicaciones web.
- **Basic Web Session Impersonation** (8) : explica los peligros de las sesiones y cómo se pueden falsear.

¿Cómo se emplean las sesiones con PHP?

- **PHP Sesiones** (9) : documentación oficial del uso de la sesiones en PHP.
- **PHP Sessions** (10) : explicación sencilla sobre el uso de las sesiones en PHP.

3. ¿Qué tengo que hacer?

En esta práctica tienes que emplear las cookies para conservar información de los usuarios entre diferentes visitas al sitio web. En concreto, tienes que emplear las cookies desde PHP para:

- Mostrar la opción de “recordarme en este equipo” en el formulario de acceso como usuario registrado en la página principal, para que almacene en una cookie el nombre de usuario y la contraseña y no se tenga que solicitar en próximas visitas. Se debe recordar al usuario durante 90 días. En la Figura 1 se muestra el empleo de esta opción en los sitios web Hotmail y Blogger.

Cuando se esté recordando a un usuario, en vez del formulario de acceso se tienen que mostrar el nombre del usuario recordado con un mensaje del estilo `Hola Pepito` y dos botones o enlaces: uno que permita acceder a la parte privada con el usuario recordado y otro que cierre la sesión, borre los datos del usuario recordado y permita acceder con otro usuario.

- Mostrar un mensaje con la fecha y la hora de la última visita al sitio web: cuando no se está recordando a un usuario no se debe mostrar ningún mensaje, pero cuando se recuerde a un usuario se debe mostrar la fecha y la hora de la última visita.

Figura 1: Opción “recordarme en este equipo” en Hotmail y Blogger

---

En la Figura 2 se muestra un ejemplo de cómo se tiene que hacer lo que se pide: a la izquierda se muestra el formulario de acceso cuando no se está recordando a un usuario, mientras que a la derecha aparece cómo se tiene que visualizar cuando se está recordando a un usuario.

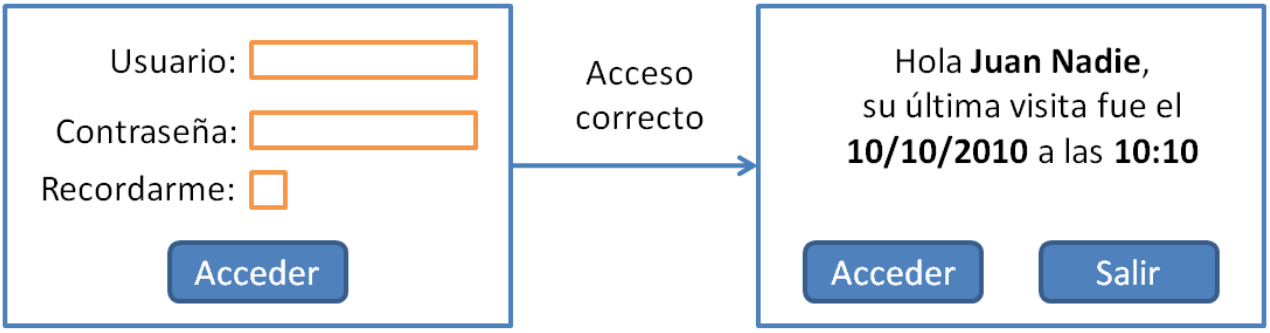

Figura 2: Cambio del formulario de acceso cuando se está recordando un usuario

---

Cuando se está recordando a un usuario, el usuario no tiene que introducir su nombre de usuario y contraseña para poder acceder a la parte privada, el acceso tiene que ser directo. En la Figura 2 se muestra un botón “Acceder”, pero este botón no es necesario si la parte privada y la parte pública de la aplicación

están integradas, solo tiene sentido cuando exista un enlace específico para acceder a la parte privada de la aplicación. Si has optado por definir la parte privada como integrada en la parte pública, entonces quizás no necesites el botón de "Acceder", simplemente muestra el nombre del usuario y los datos del último acceso.

Muy importante: se tiene que comprobar que el nombre de usuario y la contraseña almacenadas al recordar un usuario coinciden con la lista de usuarios válidos. Sólo se tiene que recordar la información de un usuario, el último usuario que haya seleccionado la opción de recordar.

Además, tienes que emplear las sesiones de PHP para controlar el correcto acceso como usuario registrado. Ahora mismo existe un problema de seguridad en el sitio web: si se conoce la URL, se puede acceder directamente a la página con el menú de usuario registrado sin tener que pasar por la página de control de acceso. Tienes que programar un sistema para que sólo se muestre la página con el menú de usuario registrado a aquellos usuarios que hayan introducido un nombre de usuario y una contraseña correctos.

Además, en la **Página menú usuario registrado** (la página principal de la parte privada) tienes que mostrar el nombre de usuario del usuario que ha accedido con un mensaje del estilo `Hola Pepito`.

Además, la **Página detalle foto**, que muestra todos los datos de una foto seleccionada en la página principal o en la página con el listado resultado de una búsqueda, sólo debe ser visible para aquellos usuarios que se hayan registrado e identificado. Los usuarios no registrados pueden ver el listado, pero no el detalle de cada foto.

En las prácticas de CSS realizaste varios estilos para tu sitio web: un estilo "clásico" sin características de diseño adaptativo (*responsive design*), un estilo adaptativo y un estilo accesible. Incluso, es posible que creases temas con variaciones de color y de tipo de letra sobre una misma hoja de estilo. Algunos navegadores permiten a través de un menú seleccionar el estilo alternativo, pero no todos los navegadores lo permiten. Además, cuando se selecciona un estilo alternativo, al pasar de una página web a otra de un mismo sitio web a través de un enlace, no se conserva la selección del estilo alternativo y se muestra siempre el estilo principal. ¿Cómo se puede resolver este problema?

En esta práctica tienes que emplear las sesiones de PHP para conservar el estilo seleccionado por el usuario. Por ahora, la selección del estilo no se puede cambiar, la tienes que almacenar con los datos de los usuarios que puedan acceder a la parte privada. En **una próxima práctica** tendrás que permitir que el usuario seleccione el estilo alternativo que desea.

Por último, en el menú de usuario registrado tienes que añadir la opción Salir (11) para que finalice la sesión del usuario e impida un acceso posterior sin antes volver a introducir un nombre de usuario y una contraseña o volver a acceder como usuario recordado en el equipo. No se tiene que implementar una opción "Olvidarme": los datos recordados se borran cuando el usuario cierra la sesión de forma explícita.

Con el fin de aclarar el funcionamiento de la práctica, a continuación se comenta el funcionamiento de Twitter (Figura 3) y Wordpress (Figura 4):

1. Al acceder al sitio web la primera vez, se muestra el formulario para iniciar sesión.
2. Si el usuario activa la opción "Recuérdame", cuando el usuario cierre el navegador web y retorne al sitio web no se volverá a mostrar el formulario para iniciar sesión.
3. Cuando el usuario cierra la sesión, se cierra la aplicación y sale de la parte privada. Si además había activado la opción "Recuérdame", la aplicación borra los datos empleados para recordar al usuario.

Figura 3: "Recordar mis datos" y "Cerrar sesión" en Twitter

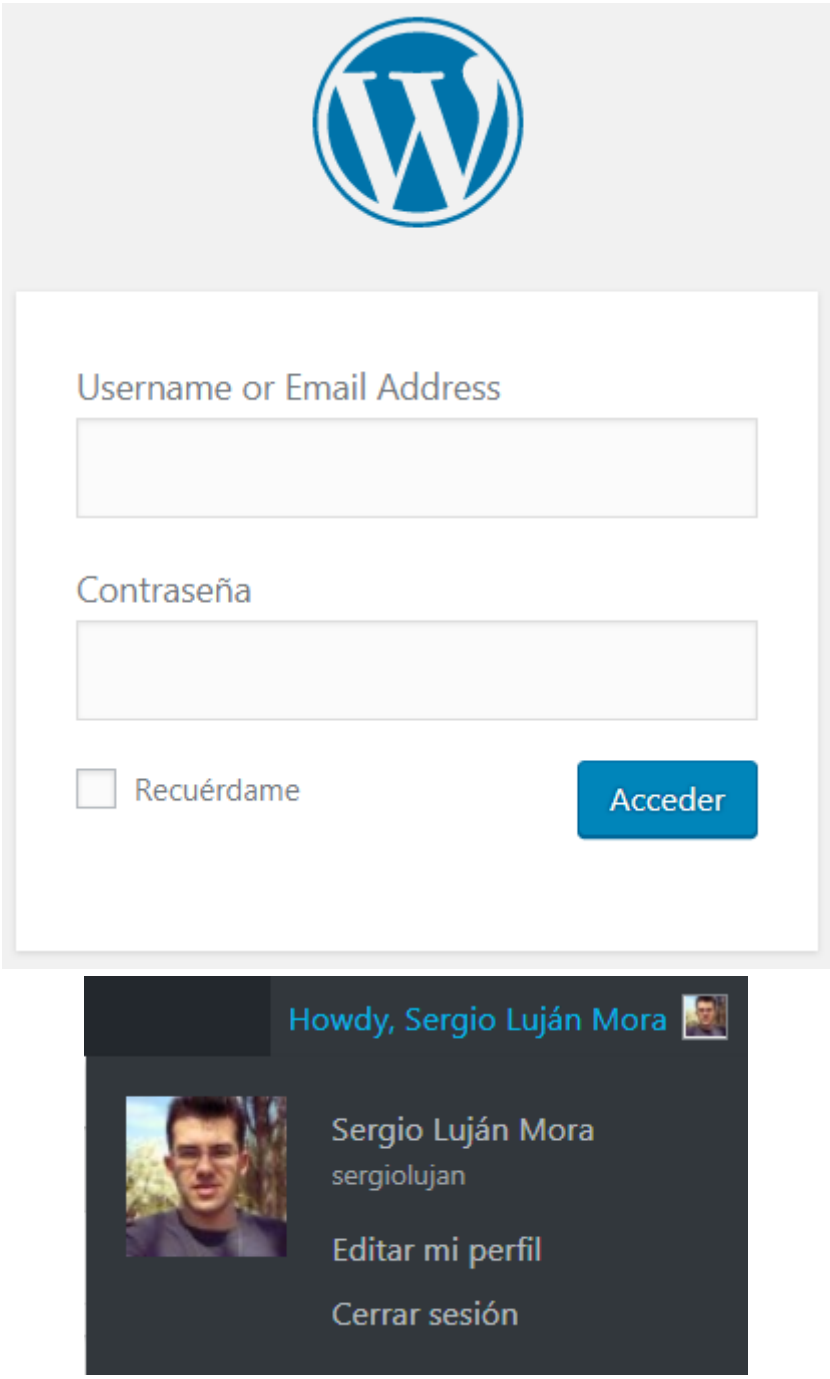

Figura 4: "Recuérdame" y "Cerrar sesión" en Wordpress

3.1. Recordatorio

La parte privada de la aplicación y su integración con la parte pública la puedes plantear de varias formas. Dos formas típicas son:

Separada
: La parte privada es completamente independiente de la parte pública, posee su propio menú o barra de navegación e incluso puede poseer su propio estilo visual (CSS). Evidentemente, debe existir una opción en el menú o barra de navegación que permita pasar de la parte pública a la parte privada y viceversa.

Integrada
: La parte privada se integra como una opción más en el menú o barra de navegación de la parte pública. La parte privada aparece como un apartado más de la parte pública, que sólo está disponible

cuando el usuario se ha identificado.

En la realización de esta práctica puedes aplicar cualquiera de estas dos estrategias o cualquiera similar.

4. ¿Cómo lo hago?

4.1. Cookies

En los primeros años del desarrollo web, la persistencia de la información asociada a un usuario conforme se movía de una página a otra era una tarea complicada que requería mucho ingenio. Dos técnicas muy extendidas eran **adjuntar los datos a la URL de la página web** o **almacenar los datos en campos ocultos de un formulario**. Esto permitía mantener los datos de una página a otra, pero una vez que se cerraba el navegador web, los datos se perdían.

La aparición de las cookies a finales de 1994 (12) supuso toda una revolución, ya que permitía la persistencia de los datos incluso si se cerraba el navegador web. Con las cookies, se podía cerrar el navegador web y volver a abrirlo otro día y aún tener acceso a los mismos datos que el día anterior.

Estas técnicas aún existen, pero ahora, debido a las necesidades de aplicaciones web más complejas, incluidas las aplicaciones web que funcionan en el navegador sin conexión a Internet, los navegadores web admiten técnicas más sofisticadas de almacenamiento de datos y de su persistencia: `sessionStorage` y `localStorage`.

Una cookie es un fragmento de información que un navegador web almacena en el disco duro del visitante a una página web. La información se almacena a petición del servidor web, ya sea directamente desde la propia página web con JavaScript o desde el servidor web mediante las cabeceras HTTP, que pueden ser generadas desde un lenguaje de web scripting como PHP. La información almacenada en una cookie puede ser recuperada por el servidor web en posteriores visitas a la misma página web.

Las cookies resuelven un grave problema del protocolo HTTP: al ser un protocolo de comunicación "sin estado" (*stateless*), no es capaz de mantener información persistente entre diferentes peticiones. Gracias a las cookies se puede compartir información entre distintas páginas de un sitio web o incluso en la misma página web pero en diferentes instantes de tiempo.

En PHP se emplea la función `setcookie()` para asignar valor a una cookie. El prototipo de esta función es:

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

Para borrar una cookie, se tiene que asignar a la cookie una fecha de caducidad (`expire`) en el pasado, es decir, una fecha anterior a la actual.

Para recuperar el valor de una cookie se emplea el array predefinido `$_COOKIE` con el nombre de la cookie como índice. También se puede emplear `$_REQUEST`, que contiene la unión de `$_COOKIE`, `$_POST` y `$_GET`.

En la siguiente página se cuenta el número de visitas que realiza un usuario al visitar la página; este contador conserva su valor durante un año aún a pesar de que un usuario cierre el navegador y tarde días en volver a visitar la página:

```
<?php
 if(isset($_COOKIE['contador']))
 {
```

```
    // Caduca en un año
    setcookie('contador', $_COOKIE['contador'] + 1, time() + 365 * 24 * 60 * 60);
    $mensaje = 'Número de visitas: ' . $_COOKIE['contador'];
  }
  else
  {
    // Caduca en un año
    setcookie('contador', 1, time() + 365 * 24 * 60 * 60);
    $mensaje = 'Bienvenido a nuestra página web';
  }
?>
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8" />
<title>Prueba de cookie</title>
</head>
<body>
<p>
<?php echo $mensaje; ?>
</p>
</body>
</html>
```

En el código anterior, fíjate cómo se emplea la función `isset()` para comprobar si una variable ha sido inicializada.

Importante: las cookies se envían al cliente mediante encabezados HTTP. Como cualquier otro encabezado, las cookies se deben enviar antes que cualquier salida que genere la página (antes que `<html>`, `<head>` o un simple espacio en blanco o salto de línea).

Sin embargo, se puede emplear un buffer de salida para almacenar (retener) la salida generada, en cuyo caso se puede enviar un encabezado en cualquier momento, pero con la sobrecarga que supone por un lado almacenar en el servidor toda la salida antes de ser enviada y por otro lado no recibir la página en el cliente hasta que el código PHP no haya terminado completamente. Esto se puede lograr con las funciones `ob_start()` y `ob_end_flush()` en la propia página, o configurando el parámetro `output_buffering` en el fichero `php.ini`.

En el manual oficial de PHP, en la entrada “Configuración en tiempo de ejecución” (13) se puede leer:

> **output_buffering boolean/integer**
>
> Se puede habilitar el búfer de salida para todos los ficheros estableciendo esta directiva a ‘On’. Si se necesita limitar el tamaño del búfer a un tamaño en particular, se puede usar un número máximo de bytes en lugar de ‘On’ como valor para esta directiva (p.ej., output_buffering=4096). A partir de PHP 4.3.5, esta direcitva siempre es ‘Off’ en PHP-CLI.

Suponer que esta directiva va a estar siempre a ‘On’ es un error. Por ejemplo, en nuestro ordenador de desarrollo puede estar activada, pero en el ordenador de producción o en un servicio de alojamiento (*hosting*) puede estar desactivada. Por ello, **lo mejor es tenerla desactivada siempre en el ordenador de desarrollo, para evitar problemas en el futuro**.

4.2. Sesiones

Una sesión es un mecanismo de programación de las tecnologías de web scripting que permite conservar información sobre un usuario al pasar de una página a otra, es decir, entre diferentes accesos a un mismo sitio web. A diferencia de una cookie, los datos asociados a una sesión se almacenan en el servidor y nunca en el cliente.

En la mayoría de las tecnologías de web scripting, las sesiones se implementan mediante una cookie que almacena un valor que identifica al usuario en el servidor web cada vez que pasa de una página web a otra. En el servidor web están almacenados todos los datos de la sesión y se accede a ellos cada vez que se pasa de página gracias al identificador almacenado en la cookie.

Importante: como las sesiones se implementan mediante cookies, se aplica lo dicho anteriormente sobre las cookies. Las sesiones (cookies) se envían al cliente mediante encabezados HTTP. Como cualquier otro encabezado, las sesiones (cookies) se deben iniciar (enviar) antes que cualquier salida que genere la página (antes que `<html>`, `<head>` o un simple espacio en blanco o salto de línea).

En Mozilla Firefox podemos ver las cookies que tenemos almacenadas, con su valor y fecha de caducidad. En la Figura 5 podemos ver la cookie `PHPSESSID` que se emplea en PHP para almacenar el identificador de una sesión. Además, el siguiente código de PHP muestra el identificador de una sesión y el nombre de la cookie que almacena la sesión:

```
<?php
 session_start();

 echo 'session_id(): ' . session_id();
 echo "<br />\n";
 echo 'session_name(): ' . session_name();
 echo "<br />\n";
 print_r(session_get_cookie_params());
?>
```

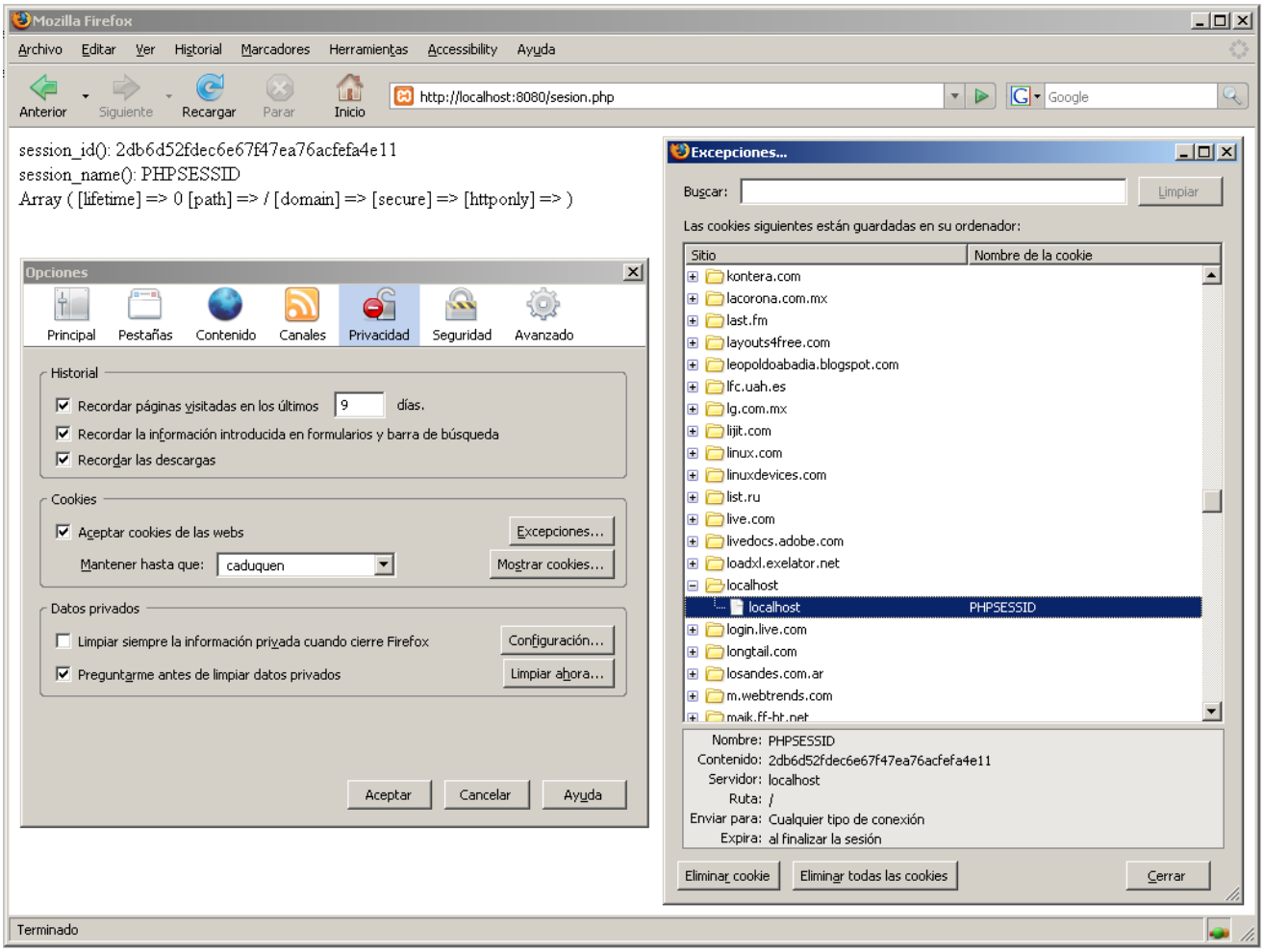

Figura 5: Cookie que almacena el identificador de sesión

El siguiente ejemplo cuenta el número de visitas que realiza un usuario a una página web. A diferencia del contador realizado con una cookie en el ejemplo anterior, este contador sólo funciona durante el tiempo de existencia de la sesión, si el usuario cierra el navegador y vuelve a acceder a la página, el contador se reiniciará desde uno:

```
<?php
 session_start();

 if(isset($_SESSION['contador']))
 {
   $_SESSION['contador'] = $_SESSION['contador'] + 1;
   $mensaje = 'Número de visitas: ' . $_SESSION['contador'];
 }
 else
 {
   $_SESSION['contador'] = 1;
   $mensaje = 'Bienvenido a nuestra página web';
 }
?>
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8" />
<title>Prueba de cookie</title>
</head>
<body>
<p>
<?php echo $mensaje; ?>
</p>
</body>
</html>
```

Para destruir una sesión y toda la información que contiene lo mejor es borrar explícitamente toda la información contenida en el array `$_SESSION`, borrar la cookie que contiene el identificador de la sesión y por último destruir la sesión con una llamada a la función `session_destroy()`:

```
<?php
 // Borra todas las variables de sesión
 $_SESSION = array();

 // Borra la cookie que almacena la sesión
 if(isset($_COOKIE[session_name()])) {
   setcookie(session_name(), '', time() - 42000, '/');
 }

 // Finalmente, destruye la sesión
 session_destroy();
?>
```

Pero la sesión también puede desaparecer por otras razones:

1. El usuario cierra el navegador; se tienen que cerrar todas las instancias del navegador, no simplemente la pestaña o la ventana de la página que contiene la sesión.
2. El desarrollador ejecuta la función `session_destroy()`, tal como se ha explicado en el ejemplo anterior.
3. El motor de PHP elimina la sesión pasado cierto tiempo; el tiempo de duración de una cookie está fijado por los parámetros de configuración `session.cookie_lifetime` (14) y `session.gc_maxlifetime` (15).
4. Un evento inesperado: se detiene el servidor web, se apaga el servidor, etc.

Para controlar el acceso a una página web de una zona privada se suele emplear una variable de sesión que se inicializa con cierto valor en la página de control de acceso; en las páginas donde se quiere controlar si el usuario tiene permiso para acceder se consulta el valor de la variable de sesión para ver si tiene el valor esperado. Si no contiene el valor esperado, lo normal es mostrar una página con un mensaje de error o redirigir al usuario a la página principal del sitio web.

5. Recomendaciones

Recuerda que las páginas que contengan código PHP tienen que tener la extensión `.php`. Si modificas alguna página web que ya tengas hecha de prácticas anteriores para añadirle código PHP, tendrás que cambiarle la extensión y corregir todos los enlaces que apunten a esa página.

Recuerda que para que el código PHP se ejecute, la página web tiene que pasar por el servidor web para que éste se la pase al intérprete de PHP. Para ello, tienes que cargar la página a través del servidor web: la URL de conexión tiene que tener la forma `http://localhost`. Si en la barra de direcciones del navegador ves escrito algo del estilo `file:///D:/...`, el código PHP no se ejecutará.

El manual de PHP te lo puedes descargar en diferentes formatos de su sitio web (16) para tenerlo siempre a mano y poder hacer las búsquedas de información rápidamente. También puedes acceder a través de Internet a la ayuda de cualquier función de PHP escribiendo el nombre de la función a continuación de la URL `http://php.net/`. Por ejemplo, `http://php.net/header` muestra la ayuda de la función `header()`.

Recuerda que debes poner al principio de la página `session_start()` en todas las páginas donde quieras utilizar las sesiones. Si no lo pones y accedes a la variable `$_SESSION` no obtendrás ningún resultado.

Recuerda que en PHP puedes emplear las funciones `include()` y `require()` para reutilizar código entre diferentes páginas:

- `require()`: si el fichero no existe, se produce un mensaje de error y finaliza la ejecución.
- `include()`: si el fichero no existe, se produce un mensaje de advertencia y continúa la ejecución.

No te líes con las cookies y las sesiones, se parecen, pero no son lo mismo:

- Las cookies permiten almacenar información de forma persistente durante un tiempo “prácticamente ilimitado” (varias decenas de años). Las cookies se almacenan en el lado del cliente, en el navegador de cada usuario.
- Las sesiones permiten almacenar información de forma persistente por un tiempo limitado: el valor por defecto del tiempo de vida de la cookie que se emplea para identificar la sesión es “0”, lo que significa “hasta que el navegador se cierre” (17) . Las sesiones se almacenan en el lado del servidor.

En la práctica no puedes emplear JavaScript, así que debes impedir que la cookie sea accesible por JavaScript, ¿lo sabes hacer?

Almacenar el nombre de usuario y la contraseña de usuario registrado directamente en una cookie es una mala práctica, ya que supone un problema de seguridad porque cualquier persona que use el ordenador donde se almacena la cookie puede tener acceso a esos datos y puede utilizarlos para suplantar la identidad del usuario y conectarse como ese usuario desde otros ordenadores. Una solución puede ser almacenar los datos encriptados o, mucho mejor, no almacenar directamente los datos, sino algún tipo de identificador (también encriptado) asociado con el usuario en el servidor. Evidentemente, sigue existiendo el peligro de que alguien suplante al usuario, pero sólo si tiene acceso a ese ordenador (18) .

Si se está empleando un ordenador compartido no se debe activar la opción de “recordarme en este equipo”, porque entonces cualquier persona que utilice el ordenador podrá acceder con el perfil del usuario almacenado. Cuando utilices un ordenador compartido, emplea la opción Limpiar información

privada... en Mozilla Firefox o Eliminar el historial de exploración... en Microsoft Internet Explorer para eliminar todo rastro de tus datos personales y de tu actividad en el navegador. O mucho mejor, utiliza la opción de "navegación privada" o "navegación de incognito" que existen en la mayoría de los navegadores web.

Pero a veces es al revés, a veces te puede interesar que se conserve el historial de navegación, pero el navegador puede estar configurado para no hacerlo. Por ejemplo, en Mozilla Firefox, en Opciones, dentro del apartado Privacidad & Seguridad está la opción Historial, tal como se puede ver en la Figura 6. Si está seleccionada la opción No recordar el historial, las cookies no se conservarán cada vez que se cierre el navegador web.

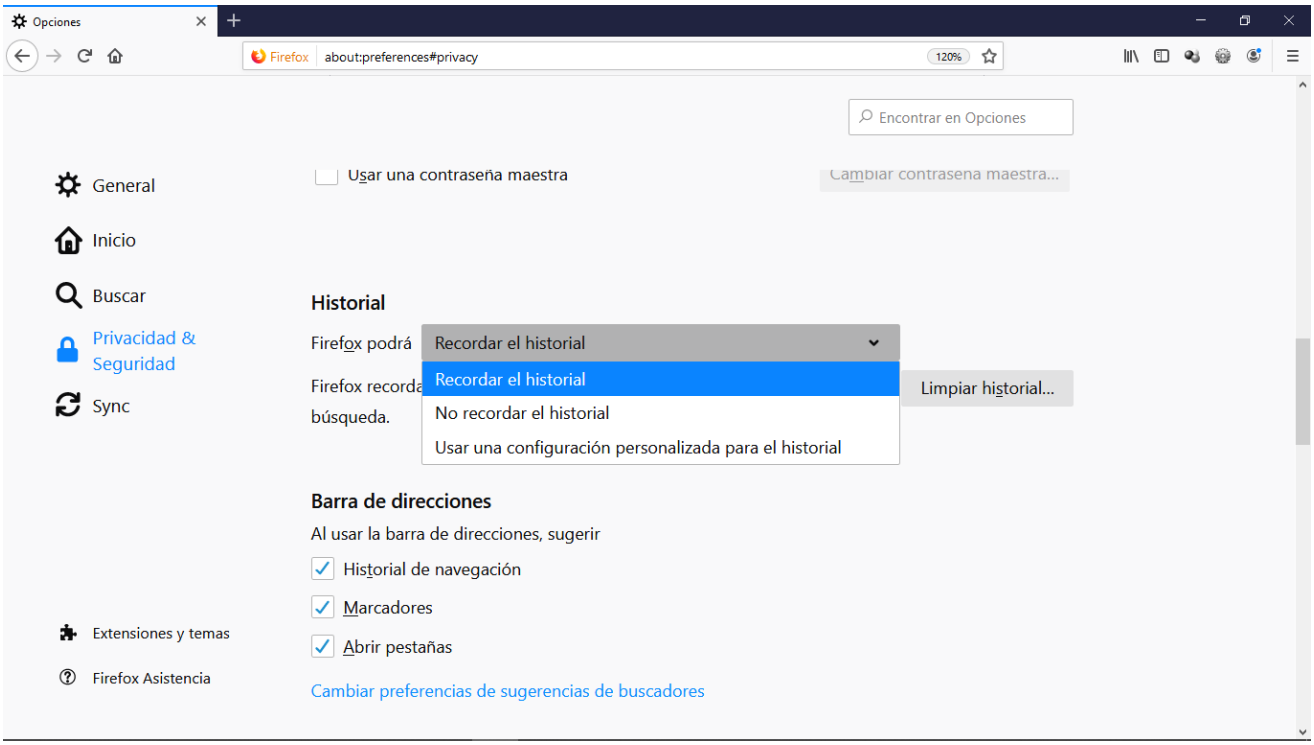

Figura 6: Configuración del historial en Mozilla Firefox

Existen diferentes opciones para consultar, modificar y eliminar las cookies que almacena un navegador web. Por ejemplo, en Google Chrome puedes emplear la extensión Web Developer (19) (también disponible para Mozilla Firefox (20)) y en Mozilla Firefox puedes emplear Cookie Manager (21) o Cookie Quick Manager (22).

La Figura 7 muestra las opciones del menú Cookies de Web Developer.

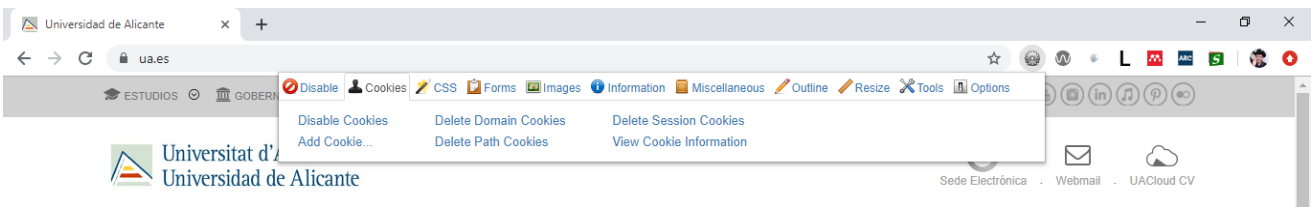

Figura 7: Menú "Cookies" de Web Developer

La Figura 8 muestra las cookies que contiene la página principal del sitio web de la Universidad de Alicante mediante la opción “View Cookie Information” de Web Developer. Para cada cookie se muestra su nombre, valor, dominio y ruta en la que existe, fecha de caducidad y los *flags* de segura y solo disponible en el servidor. Se puede ver que hay un botón para eliminar (*Delete*) la cookie y otro para modificarla (*Edit*).

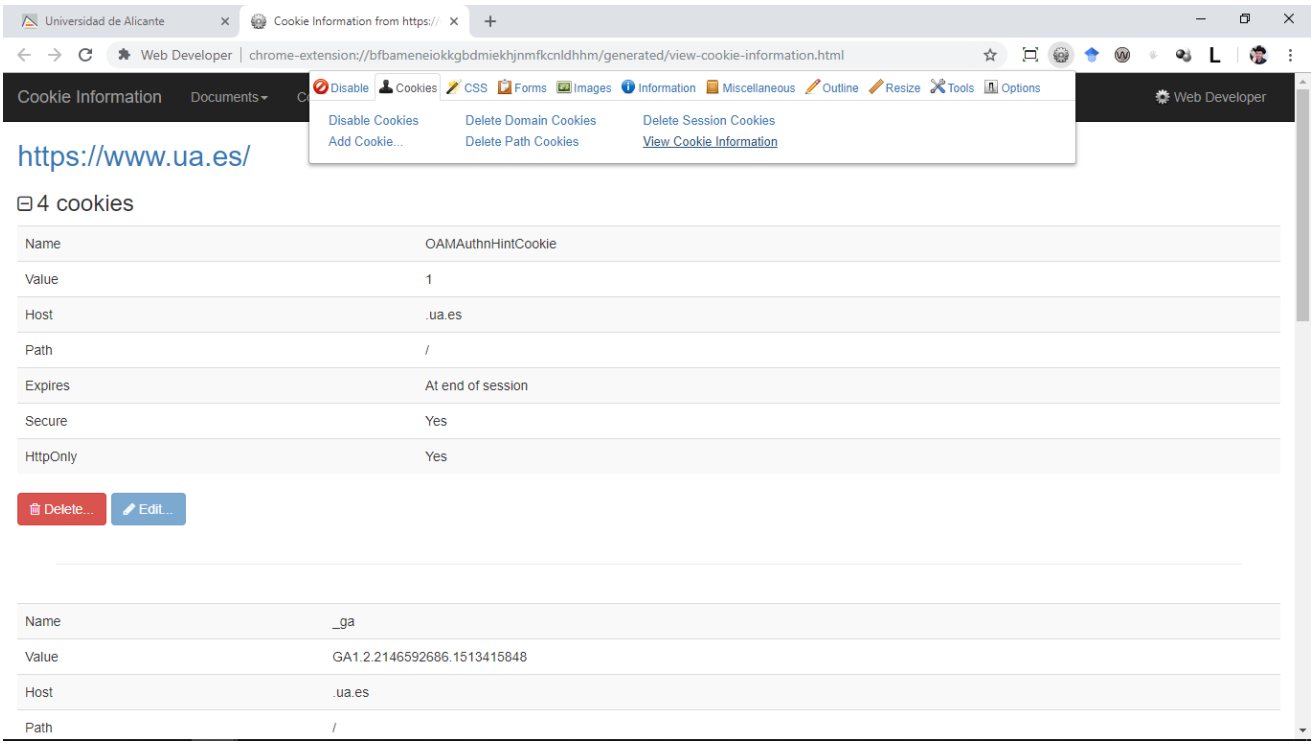

Figura 8: “View Cookie Information” de Web Developer

La Figura 9 muestra las cookies que contiene el navegador web Mozilla Firefox mediante Cookie Quick Manager. En el panel de la izquierda Domains aparecen todos los sitios web que han almacenado alguna cookie. Esta extensión también permite modificar y eliminar las cookies almacenadas.

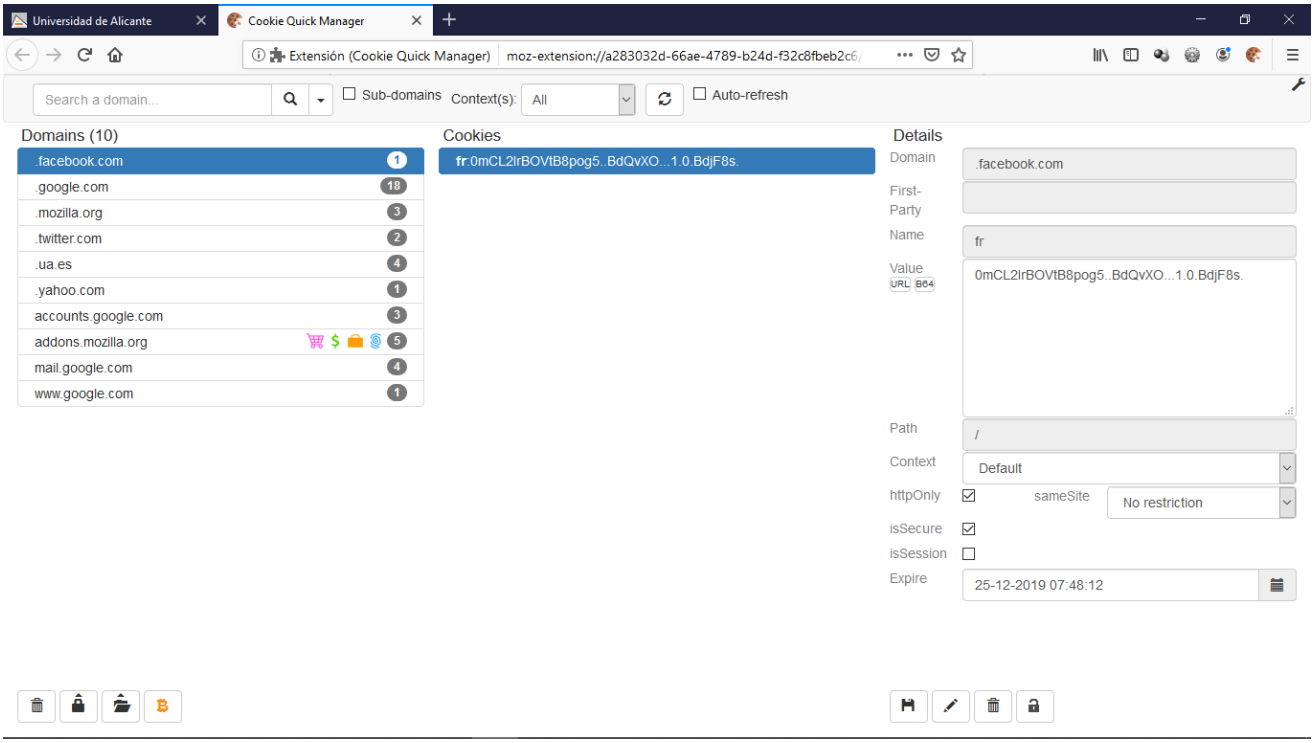

Figura 9: Cookie Quick Manager

---

(1) http://tools.ietf.org/html/rfc2965

(2) http://es.wikipedia.org/wiki/Cookie

(3) http://www.cookiecentral.com/c_concept.htm

(4) http://www.php.net/manual/es/features.cookies.php

(5) http://www.w3schools.com/PHP/php_cookies.asp

(6) http://www.ignside.net/man/php/cookies.php

(7) http://livedocs.adobe.com/dreamweaver/8_es/using/wwhelp/wwhimpl/common/html/wwhelp.htm?context=LiveDocs_Parts\&file=34_obt10.htm

(8) http://www.securityfocus.com/infocus/1774

(9) http://www.php.net/manual/es/features.sessions.php

(10) http://www.w3schools.com/php/php_sessions.asp

(11) O Cerrar sesión*, como la quieras llamar.*

(12) En octubre de 1995 se presentó la solicitud de patente US5774670 "Persistent client state in a hypertext transfer protocol based client-server system" en la que se presentaba la idea de las cookies. La patente fue concedida en junio de 1998.

(13) https://www.php.net/manual/es/outcontrol.configuration.php

(14) http://php.net/manual/es/session.configuration.php#ini.session.cookie-lifetime

(15) http://www.php.net/manual/en/session.configuration.php#ini.session.gc-maxlifetime

(16) http://www.php.net/download-docs.php

(17) http://www.php.net/manual/es/session.configuration.php\#ini.session.cookie-lifetime

(18) En la información almacenada en la cookie, también conviene almacenar información asociada al ordenador para evitar que la cookie sea robada y utilizada desde otro ordenador.

(19) https://chrome.google.com/webstore/detail/web-developer/bfbameneiokkgbdmiekhjnmfkcnldhhm?hl=es

(20) https://addons.mozilla.org/es/firefox/addon/web-developer/

(21) https://addons.mozilla.org/es/firefox/addon/a-cookie-manager/

(22) https://addons.mozilla.org/es/firefox/addon/cookie-quick-manager/?src=search