-

UD04. GESTIÓN DE SESIONES EN PHP

Desarrollo web entorno servidor
CFGS DAW

Autor: Francisco Aldarias Raya - paco.aldarias@ceedcv.es

Revisado por: Vanessa Tarí – vanessa.tari@ceedcv.es

2020/2021

Versión:041120.1135

Licencia

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

Importante

Atención

Interesante

Código

Revisiones.

- 04/11/20 – Añadido campo seguridad en cookies por Vanessa Tarí

INDICE

1. MANTENIMIENTO DEL ESTADO EN APLICACIONES WEB 4
2. COOKIES 5
2.1 Funcionamiento de las cookies 6
2.2 Crear cookies 9
2.3 Consultar valor de la cookie 11
2.4 Caducidad de las cookies 14
2.5 Modificar cookies 16
2.6 Borrar cookies 16
2.7 Limitaciones de las cookies 16
2.7.1 Limitación en tamaño 17
2.7.2 Limitación en juego de caracteres 17
2.7.3 Dependencia del navegador 17
3. SESIONES 18
3.1 Crear sesiones 19
3.2 Consultar las sesión 20
3.3 Borrado de una variable de la sesión 20
3.4 Destruir la sesión 21
4. BIBLIOGRAFÍA 22

UD04. GESTIÓN DE SESIONES

1. MANTENIMIENTO DEL ESTADO EN APLICACIONES WEB

El mantenimiento de los datos es fundamental para gestionar la comunicación entre usuarios y servidor. Si queremos acumular la cesta de la compra en una página web, actualizar una base de datos o controlar el acceso de usuarios deberemos mantener el estado de la conexión entre páginas o visitas al sitio.

Sin embargo, la Web se basa en un modelo *stateless* (no conectado), ya que el protocolo HTTP es un protocolo sin estado. Esto quiere decir que no permite mantener el estado en una aplicación web, es decir, saltar de una página a otra implica la pérdida de datos y no guarda los estados de las mismas. De hecho, en cada petición al servidor se vuelve a crear toda la página, por lo que los datos se pierden entre llamadas. Y esto es justamente uno de los requisitos principales para gestionar la interoperabilidad entre los usuarios y un sistema online (por ejemplo, una tienda online).

De este modo, es necesario crear mecanismos que permitan guardar información entre las diferentes llamadas al servidor o actualizaciones de la página dinámica. La única forma de garantizar la permanencia de dichos estados es a través de un **mecanismo de sesiones**, que permita registrarlos haciéndolos consistentes y gestionables.

El manejo de sesiones nos permite controlar la ejecución correcta y segura de los recursos o servicios ofrecidos por la aplicación web. El objetivo de una **sesión** *es* «*guardar información específica de cada usuario mientras este navega por una aplicación web*» (López, 2012: 126). Así, cada usuario que entra en un sitio web abre una sesión diferente e independiente de la de los otros usuarios. La sesión evita problemas de inconsistencia (ej. que el usuario pase a otra página y pierda información o no pueda seguir con la operación *online*) y también problemas con el servidor (perdería el control sobre lo que ocurre hasta el punto de dar acceso a un usuario no autorizado o a la inversa).

☐ Interesante: Las funciones de sesión se pueden consultar en: https://www.php.net/manual/es/ref.session.php

Para establecer una sesión entre el servidor de una aplicación web y el cliente, se puede desarrollar un **mecanismo de control** de dos tipos:

- *Enviando las sesiones a través de cookies*: es la nueva técnica que vamos a analizar en este tema
- *Enviando las sesiones a través de URL*: se pasan los datos con técnicas GET/POST como hemos visto hasta ahora.

Cuando un cliente se identifica en el servidor, el servidor envía al cliente un **código único de identificación** y gestión de sesión llamado **SID**. Éste es un número asignado desde el servidor aleatoriamente y se usa para gestionar la sesión. El SID se almacena en el cliente temporalmente hasta que abandona el recurso cerrando la sesión en el servidor. El recurso físico del cliente (ordenador) sólo guarda este número de identificación. En cambio, en el lado del servidor se guarda el número más otras variables de la sesión y gestión interna.

☐ Interesante

Envío de las sesiones a través de URL:

* Si se hace con GET los datos se incluyen en la dirección que apunta al recurso. Si se usa POST, los datos se anexan en el encabezado de la solicitud de envío que se dirige al recurso apuntado.
* Con un QueryString se almacenan los valores en la URL siguiendo el esquema clave=valor, separando cada una de estas parejas con el símbolo ampersand &.
* Este método de transferencia tiene como problemas la capacidad limitada del espacio disponible (1024 bytes), con lo que no es posible enviar objetos complejos. Además, es un método no seguro al viajar la información de manera visible.
* Los valores almacenados se pierden al escribir una nueva dirección URL o al cerrar el navegador.

2. COOKIES

Las cookies o *galletas de información* son un mecanismo que utilizan los servidores web para guardar información en el ordenador del usuario y recuperarla cada vez que el navegador les pide una página. Las cookies permiten acceder a la información almacenada entre diferentes peticiones al servidor incluso con varios días de diferencia. Debido al formato del archivo, este tipo de almacenamiento permite ser consultado por el propio usuario una vez es localizado en su disco duro.

La información se guarda en el ordenador del usuario en forma de texto y está formada por parejas (nombre de la cookie y valor de la cookie). Así, en el cliente se almacena un pequeño archivo que guarda el número SID y otros parámetros (expiración, páginas dirigidas, recursos apuntados...).

Esta información se utiliza para numerosos fines (autentificación, selección de preferencias, ítems seleccionados en un carrito de la compra, etc.), siempre con la intención de identificar al usuario y personalizar las páginas.

☐ Interesante

En el manual de PHP se ofrece un capítulo dedicado a las cookies: http://www.php.net/manual/es/features.cookies.php

La característica fundamental de utilizar cookies es el hecho de que dicha información se almacena en el ordenador cliente, es decir, es responsabilidad del navegador. Esto permite delegar dicha responsabilidad, liberando al servidor de la tarea de mantener esa información.

Sin embargo, aunque esto es una ventaja, también se corre el riesgo de que el usuario modifique dicha información o incluso que la elimine. Por tanto, el problema de este mecanismo de almacenamiento es la privacidad. Es habitual encontrar usuarios que deniegan el uso de cookies en su navegador, o incluso las borran, lo que imposibilita su uso en aplicaciones web. Por ello, un sitio dirigido al comercio electrónico tiene que estar preparado para esta circunstancia.

Para evitar que este inconveniente condicione la funcionalidad del sistema, algunos lenguajes embebidos como PHP usan un **sistema que fuerza a usar ambos sistemas** (basados en URL o en cookies). Así, se garantiza que en caso de no usar cookies, el sistema siga operando usando URL por defecto como última alternativa. Esta opción se configura en el archivo de configuración php.ini.

El manejo de las cookies consta de:

- Crear cookies
- Modificar cookies
- Borrar cookies
- Utilizar cookies

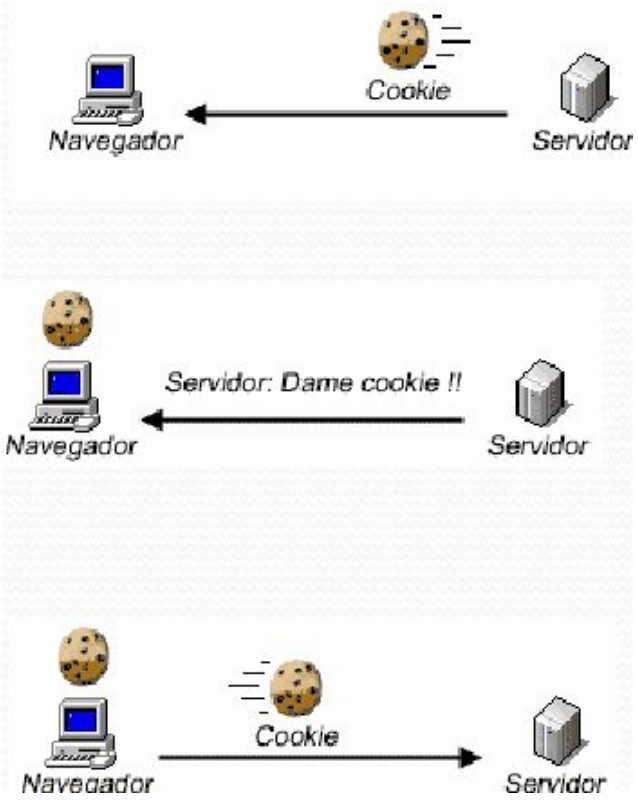
Veamos en detalle cada uno de ellos.

2.1 Funcionamiento de las cookies

Las cookies se crean cuando el servidor se lo pide al navegador. Cuando el servidor envía una página al navegador, puede incluir en las cabeceras de la respuesta HTTP la petición de creación de una o varias cookies. El navegador crea la cookie, guardando no sólo el nombre y el valor de la cookie, sino el nombre del servidor (del dominio) que ha creado la cookie.

Cuando se inicia por primera vez una sesión, el servidor envía una petición de cookie al navegador del usuario y cada vez que el navegador del usuario solicite una página de ese servidor, se volverá a enviar la cookie al servidor, el cual puede leer la cookie e identificar el navegador del usuario.

Realmente las cookies no están relacionadas con una página o un servidor web específico sino con un determinado dominio. Por ejemplo, accediendo al sitio web *www.elcorteingles.es* , seguramente tendremos en nuestro equipo una cookie relacionada con el dominio *elcorteingles.es*.



Para crear cookies, se utiliza la función **setcookie()**, mientras que para acceder a la información almacenada en las cookies, utilizaremos el *array* superglobal **$_COOKIE**. Esta función crea un pequeño archivo de texto que tiene el número de identificación de la sesión (SID).

Entre otras utilidades, escribiremos código para crear cookies que nos permitan recordar al usuario que accede a la página cuándo fue la última vez que la visitó, esto suele aparecer en sitos seguros, como banca electrónica, donde es importante recordar cuándo se accedió por última vez.

Hay que tener la precaución de **utilizar la función *setcookie()* antes de empezar a escribir el contenido de la página**, porque si no PHP producirá un aviso y no se creará la cookie. El motivo es que las cookies se crean mediante cabeceras de respuesta HTTP y las cabeceras se envían antes del texto de la página. Es decir, cuando PHP encuentra una instrucción que escribe texto, cierra automáticamente la cabecera; si a continuación PHP encuentra en el programa la función *setcookie(),* da un aviso porque ya se han enviado las cabeceras y no se crea la cookie. El ejemplo siguiente muestra código incorrecto, ya que utiliza la función *setcookie()* después de haber escrito texto, y el mensaje de aviso generado por PHP.

```php
<?php
// Este código es incorrecto, la cabecera se crea después de crear texto.
// Debe estar antes el setcookie. print "<p>Hola</p>";
setcookie("nombre", "Pepito Grillo");
?>
```

Se genera este mensaje:

El Warning: Cannot modify header information - headers already sent by (output started at cookiecabecera.php:3) in cookiecabecera.php on line 4

En algunos casos este código incorrecto puede no generar un aviso y la cookie puede crearse, dependiendo de la configuración de la directiva output_buffering en php.ini.

Es muy importante que **los nombres de las cookies no coincidan con los nombres de controles de los formularios**, porque PHP incluye los valores de las cookies en la matriz $_REQUEST. Es decir, que si el nombre de una cookie coincide con el nombre de un control, en $_REQUEST sólo se guardará el valor de la cookie, no el del control.

Ejemplo: En el siguiente ejemplo se muestra en pantalla el nombre de la cookie y no el del control ya que se llaman igual.

```
<?php
setcookie("nombre", "juan");
if (isset($_REQUEST ["nombre"])) {
      echo "Nombre Cokiee: " . $_REQUEST["nombre"];
      // Se muestra el cookie y no el control si se llaman igual
}
?>
<html>
<form action="" type="get">
Nombre: <input type="text" name="nombre" value="paco"/>
<input type="submit" name= "Enviar" value="Enviar">
<input type="reset" name= "Borrar" value="Borrar">
</form>
</html>
```

Resultado:

La creación de cookies tiene límites, pero cada navegador tiene un límite distinto. En muchas páginas web se puede leer que el límite por dominio suele ser de 20 cookies (si se hacen más, se borran las más antiguas), que el límite de tamaño del valor almacenado suele ser de 4096 bytes y que el límite del número total de cookies suele ser de 300 cookies, pero estos valores pueden ser distintos en versiones más recientes.

En resumen, el manejo de las cookies en PHP es muy sencillo:

* Se envía la cookie (hemos de llamar a setcookie() para crear la cookie en el cliente web)

* En las posteriores peticiones que se reciban de ese cliente ya vendrá incrustada la cookie, con lo que únicamente se consulta la información que tiene (con $_COOKIE)
* Recordemos que las cookies deben enviarse antes que cualquier otra cabecera de HTML porque se envían en las cabeceras de las transacciones de HTTP.

2.2 Crear cookies

Importante

La función setcookie() envia cookie. Puede tener hasta siete argumentos:

setcookie($nombre, $valor, $expiracion, $ruta, $dominio, $seguridad, $solohttp)

El primero es el nombre de la cookie y suele usarse para construir el archivo cookie. El resto de los parámetros son opcionales y se usan en casos donde se requiere un comportamiento específico para las sesiones entre cliente y servidor. Un ejemplo habitual es el control del tiempo de la sesión creada:

```
<?php
$value = "algo de algún lugar";
//Una cookie sin parámetros: setcookie("TestCookie");
//Una cookie que expira en 1h: setcookie("TestCookie",$value,time()+3600);
?>
```

El parámetro *$expiración* representa el tiempo de vida de la cookie. Si hablamos de sesiones, podría ser el ciclo máximo de la sesión. Si se ajusta una sesión de 24h, la cookie hará funcionar el control del acceso por sólo 24h. Pasado el tiempo, la cookie pierde vigencia y el servidor descarta el proceso de negociación. Ésta puede ser una medida de seguridad operativa y de regulación del consumo de ancho de banda en la red.

En cuanto al parámetro *$ruta*, es el *path* desde el cual la cookie puede ser recuperada. Si se especifica el valor de la ruta como “/”, cualquier script en el servidor podrá recuperar su valor. Si especificamos un directorio (por ejemplo “/app/”), la cookie sólo podrá ser recuperada por scripts en ese directorio o subdirectorios.

El parámetro $*dominio* permite limitar el acceso a los subdominios que tengamos. Por ejemplo, si le damos el valor “miapp.net”, la cookie podrá ser recuperada tanto desde “http://miapp.net” como desde “www.miapp.net” o “dev.miapp.net”.

El campo *$seguridad* Indica que la cookie sólo debiera transmitirse por una conexión segura HTTPS desde el cliente. Cuando se configura como TRUE, la cookie sólo se creará si es que existe una conexión segura. Del lado del servidor, depende del programador el enviar este tipo de cookies solamente a través de conexiones seguras (por ejemplo, con $_SERVER[“HTTPS”]).

El parámetro *$solohttp* indica si la cookie puede ser enviada a través de una conexión segura (HTPPS) si su valor es 1 (true), o cualquier tipo de conexión (HTTP) si su valor es 0 (false).

Veamos algunos ejemplos:

```
<?php
setcookie("test1", "Pepito Sapo", time()+60); //se borrará 1 minuto
después de crearla setcookie("test2", "Pepito Pez", time()+60*60*24*365);
     //se borrará dentro de 1año setcookie("test4[animal]", "nutria");
//se reciben los datos en un array setcookie("test4[animal]", "anguila");
     //segundo dato del array

?>
```

Veamos otro ejemplo:

```
<?php
setcookie("visitas", "1", time()+86400, "", "", 1);
?>
```

En este caso estamos creando una cookie llamada *visitas*, con valor inicial 1, de duración 24 h a partir del momento en que se crea.

Hay que tener en cuenta que **la cookie no estará disponible hasta que el usuario no recargue la página**.

2.3 Consultar valor de la cookie

Las cookies son recuperadas automáticamente por el script cada vez que se carga una página, ya que éstas son enviadas como cabeceras HTTP por parte del servidor. Desde la versión 4.1 de PHP se pueden recuperar las cookies con la función S_COOKIE, que es un vector accesible desde cualquier parte del script PHP.

> Importante: La sintaxis para recuperar una cookie es:
>
> $_COOKIE[“nombre_cookie”]

Veamos un ejemplo:

```
<?php
//Mostramos el valor de una cookie llamada visitas echo
```

```
$_COOKIE["visitas"];
?>
```

Para borrar una cookie, sólo tenemos que inicializar otra cookie con el mismo nombre, pero sin ningún parámetro, por ejemplo:

```
<?php
//Borra la cookie visitas
setcookie("visitas");
?>
```

Pero el sistema a veces falla (por ejemplo, algunos navegadores no borran la cookie si no coincide con el *path*). Así que la forma más segura de borrar una cookie es colocar otra nueva, con el mismo nombre, sin valor, mismo *path* y nombre de dominio (si se especificaron) y con un tiempo de vida negativo:

```
<?php
//Borra la cookie visitas definitivamente
 setcookie("visitas", "", time()-3600);
?>
```

Cuando el navegador solicita una página PHP a un servidor (un dominio) que ha guardado previamente cookies en ese ordenador, el navegador incluye en la cabecera de la petición HTTP todas las cookies (el nombre y el valor) creadas anteriormente por ese servidor.

El programa PHP recibe los nombres y valores de las cookies y se guardan automáticamente en la matriz.

El ejemplo siguiente saluda al usuario por su nombre si el nombre del usuario estaba guardado en una cookie.

```
<?php
if (isset($_COOKIE["nombre"])) {
print "<p>Su nombre es $_COOKIE[nombre]</p>";
} else {
print "<p>No sé su nombre.</p>";
}
?>
```

En caso de que se haya guardado una matriz en forma de cookies, el ejemplo sería el siguiente:

```
<?php
if (isset($_COOKIE["datos"]["nombre"]) &&
isset($_COOKIE["datos"]["apellidos"])) {
      print "<p>Su nombre es ".$_COOKIE["datos"]["nombre"]."
".$_COOKIE["datos"] ["apellidos"]."</p>";
```

```
} else {
      print "<p>No sé su nombre.</p>";
}
```

Un detalle importante a tener en cuenta al trabajar con cookies es el orden en que se realiza el envío y la creación de cookies, así como su disponibilidad en $_COOKIES:

- cuando una página pide al navegador que cree una cookie, el valor de la cookie no está disponible en $_COOKIE en esa página.
- el valor de la cookie estará disponible en $_COOKIE en páginas posteriores, cuando el navegador las pida y envíe el valor de la cookie en la petición.

Por ello, el siguiente programa no dará el mismo resultado cuando se ejecute por primera vez que las veces posteriores:

```
<?php
setcookie("nombre", "Pepito Grillo");

if (isset($_COOKIE["nombre"])) {
      print "<p>Su nombre es $_COOKIE[nombre]</p>";
} else {
      print "<p>No sé su nombre.</p>";
}
```

La primera vez que se ejecuta este programa, ocurren las siguientes cosas en este orden:

- el navegador pide la página, pero no envía con la petición el valor de ninguna cookie, porque la cookie todavía no existe
- el servidor envía la página:
    - en la cabecera de respuesta, el servidor incluye la petición de creación de la cookie.
    - en la página escribe "No sé su nombre" porque no ha recibido ninguna cookie del navegador.

La segunda vez (y las siguientes) que se ejecuta este programa, ocurren las siguientes cosas en este orden:

- el navegador pide la página y envía con la petición el valor de la cookie "nombre".
- el servidor envía la página:
    - en la cabecera de respuesta, el servidor incluye la petición de creación de la cookie (como es el mismo valor, la cookie se queda igual).
    - en la página escribe "Su nombre es Pepito Grillo" porque ha recibido la cookie del navegador.

2.4 Caducidad de las cookies

La característica más importante de una cookie es su **caducidad**. Al crearla podemos establecer un valor entero que indique cuánto durará. Si no se indica ese valor en la función *setcookie*, la cookie se mantiene "viva" sólo en la memoria del navegador por lo que desaparece al cerrarlo. En cuanto determinamos su caducidad, las cookies se guardarán como un archivo de texto en el ordenador del usuario al que se podrá acceder mientras no caduquen.

Vamos a ver el siguiente ejemplo, donde se usan dos cookies, una de ellas almacena el número de veces que el usuario entra en la página y la otra almacena dos valores (array), la fecha y la hora de la última entrada:

```
<?php
if (empty($_COOKIE["Veces"])) {
      setcookie("Veces", 1, time() + (3600 * 24 * 7));
} else {
      setcookie("Veces", $_COOKIE["Veces"] + 1, time() +(3600 * 24 * 7));
      $Fecha_anterior = $_COOKIE["Momento"]["Fecha"];
      $Hora_anterior = $_COOKIE["Momento"]["Hora"];
}
$fecha = getdate(time());
$dia = $fecha["mday"] . "/" . $fecha["mon"] . "/" .$fecha["year"];
$hora = $fecha["hours"] . ":" . $fecha["minutes"] . ":".
$fecha["seconds"];
setcookie("Momento[Fecha]", $dia, time() + (3600 * 24 * 7));
setcookie("Momento[Hora]", $hora, time() + (3600 * 24 * 7));
?>

<html>

<head>
      <title>cookies.php</title>
      <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1">
</head>

<body>
      <?php
      if (isset($Fecha_anterior)) {
            echo "Usted visitó esta página por última vez
      el<B>$Fecha_anterior</B> a las
      <B>$Hora_anterior</B>";
            echo "<BR>Ha visitado esta página un total de: <B>"
.$_COOKIE["Veces"] . "</B> veces.";
      } else {
            echo "Bienvenido a nuestra página web.";
      }
```

```
?>
</body>

</html>
```

Aquí utilizamos la función isset para saber si la variable $Fecha_anterior tiene valor (es lo contrario que la función empty), y, en ese caso, se imprime en pantalla la fecha y hora de la última visita del usuario, además del número de veces que la ha visitado. Tal como tenemos ahora la página,
obtendríamos la primera vez el mensaje de bienvenida y, a partir de la segunda vez que se accede a la página, se nos indicaría la fecha y hora del último acceso, además del número de veces realizado.

A este último respecto, es un error habitual pensar que podemos acceder al valor actualizado de un cookie la misma vez que se modifica. Esto no es así, podremos acceder al valor que establecemos en nuestro código PHP de una cookie la siguiente vez que la página se muestre en el navegador. Sin embargo, como se trata de cookies temporales (almacenadas en la memoria del navegador), al cerrar el navegador se perderán.

Como hemos dicho, esta cookie se perderá al cerrar el navegador. Vamos a establecer una caducidad cookie. Para ello, como hemos dicho antes, en nuestro código añadiremos un parámetro más en la función setcookie() (debemos indicar la marca de tiempo correspondiente a esa fecha):

```
setcookie("Veces", 1, time() + (3600 * 24 * 7)); //vida de ahora más 7
días en segundos
```

Con esta expresión indicamos que la cookie tiene un período de vigencia de 7 días, ya que time() devuelve el momento o marca horaria actual.

Al establecer la caducidad de las cookies, éstas se almacenarán como un archivo de texto en el ordenador del usuario y de allí se obtendrán mientras no hayan caducado.

2.5 Modificar cookies

Como ya sabemos, para modificar una cookie ya existente, simplemente se debe volver a crear la cookie con el nuevo valor.

2.6 Borrar cookies

Como sabemos también, para borrar una cookie, simplemente se debe volver a crear la cookie con un tiempo de expiración anterior al presente.

Ejemplo

```
<?php
setcookie("nombre", "Pepito Grillo", time()-60); // Esta cookie se borrará
```

```
inmediatamente.
?>
```

Si solamente queremos borrar el valor almacenado en la cookie sin borrar la propia cookie, simplemente se debe volver a crear la cookie, sin indicarle el valor a almacenar:

```
<?php
setcookie("nombre"); // Esta cookie no se borra, pero no guardará ningún valor.
?>
```

2.7 Limitaciones de las cookies

Las cookies no son la mejor forma de almacenar información de estado, principalmente porque se hace en el ordenador del usuario, donde no podemos tener el control necesario. Sin embargo, este método se utiliza por su sencillez. Veamos sus principales problemáticas.

2.7.1 Limitación en tamaño

El tamaño de una cookie es una de las limitaciones existentes. Una cookie no debería sobrepasar los 4kB, sin embargo, una cookie de más de 2kB ya puede dar problemas.

Además de la limitación en tamaño, hay que contar con la cantidad de cookies posibles en el disco duro del cliente el cual, que no deberían de superar los 20 archivos. Esto se debe a que el propio sistema del cliente no puede memorizar tantos enlaces a la vez.

Se trata de una limitación implementada por razones de seguridad funcional. Si se usa algún tipo de cifrado para un valor y lo almacena dentro de la cookie, también hay que tener mucho cuidado. La cantidad de caracteres que genera un cifrador (un algoritmo de encriptación o similar), puede ser muy grande y superar la cuota de tamaño dentro de la cookie, lo que hará que se trunque la información. Por lo tanto, es recomendable evitar inflar demasiado las cookies.

2.7.2 Limitación en juego de caracteres

Otro problema es el código o juego de caracteres. El uso de un juego apropiado para un determinado país puede ser perjudicial para otro. Es por ello, que es recomendable usar un juego de caracteres estandarizado, por ejemplo, la norma ANSI'94. Esta norma sacrifica ciertos tipos de caracteres a costa de no fallar en la máquina de los clientes.

2.7.3 Dependencia del navegador

El último gran inconveniente es el bloqueo de las cookies por parte de los navegadores. El bloqueo se suele usar como medida preventiva o de seguridad.

Los principales navegadores web (Internet Explorer, Mozilla o Chrome) introducen herramientas de desarrollo web orientadas al análisis del tráfico de la red, consulta del DOM del documento web

visualizado, herramientas de depuración del código Javascript... Dependiendo del navegador usado, la información facilitada por las herramientas de desarrollo web se organizan de una forma concreta siguiendo las directrices del fabricante. Sin embargo, todos ellos aplican un enfoque común respecto de la información y funcionalidad ofrecida, siendo prácticamente idénticos. La elección de un navegador u otro dependerá de las necesidades del usuario, preferencias o restricciones del entorno.

Las cookies que se almacenan en memoria sólo están disponibles para la instancia del navegador en la que se han creado. En el caso de las cookies guardadas en disco duro, todas las instancias del navegador pueden acceder a ellas. Como existe la posibilidad de que las cookies se almacenen en el disco duro del usuario, el navegador permite que sea el usuario quien decida si esto es posible o no. (NOTA: La configuración predeterminada de Internet Explorer es Media, en la cual se permiten las cookies del sitio web pero no las de terceros.)

Con cookies de terceros se hace referencia a aquellas cookies que no provienen del dominio correspondiente al sitio web al que se está accediendo. Esto es frecuente en Internet, donde nos podemos encontrar con ventanas emergentes que aparecen, sin que lo hayamos solicitado, en muchas páginas web. Estas ventanas presentan anuncios, promociones, etc., pero también tienen el objetivo de poder alojar cookies en nuestro equipo. (NOTA: La configuración de Internet Explorer respecto de estas cookies es de no permitirlas).

A medida que configuramos una política de privacidad más estricta, menos cookies serán aceptadas por el navegador (Internet Explorer en el ejemplo). En el caso extremo no se aceptaría ninguna. También puede establecer la política de admitir o no cookies desde el cuadro de diálogo que aparece al pulsar en Opciones avanzadas, pero sólo afectará a las nuevas cookies.

En resumen, si nuestro sitio web utiliza cookies, existe la posibilidad de que las aplicaciones no funcionen bien, ya que el usuario puede haber decidido desactivar esta característica.

Pero esto no es todo. Cada navegador guarda las cookies de forma diferente, por lo que podría darse el caso de que un usuario accediera a un sitio web a través de un navegador y utilizara un navegador distinto al día siguiente: el último navegador no haría uso de las cookies almacenados en el equipo.

3. SESIONES

Hemos visto que podemos utilizar las cookies para almacenar información del usuario. La idea es proporcionar un método para que el servidor pueda diferenciar entre las distintas peticiones que realizan los usuarios. Por motivos de seguridad, almacenamiento o complejidad, no es lógico guardar información en el cliente, por lo que es necesario usar otro tipo de elementos en el servidor.

La pregunta que se plantea es: ¿cómo consigue el servidor diferenciar entre cada una de estas peticiones HTTP? Es decir, ¿cómo se identifica el navegador a la hora de realizar sus peticiones al servidor?

Para conseguirlo, el servidor trabaja con el concepto de sesión, facilitando al programador un recipiente o lugar donde almacenar información que esté disponible en toda la visita del usuario al

sitio web. Por tanto, «las sesiones son otra forma de almacenar información de un determinado usuario, pero, a diferencia de las cookies, esa información se almacena en el servidor, por lo que se tiene control total sobre dicha información»

Para cada sesión que se establezca, el servidor crea el array superglobal $_SESSION, que se mantiene hasta que se finaliza la sesión. Ésta se almacena físicamente en el servidor y en ella podremos almacenar información que queremos que esté disponible en todas las páginas de la aplicación, pero sólo para la sesión particular que la ha originado.

En conclusión, para mantener la información sobre las sesiones se utilizan las cookies. Esto implica que el navegador debe permitirlo. Si no es así, el servidor buscará maneras alternativas de transmitir la información del identificador de sesión (bien con GET o con POST).

Es menester destacar que la cookie SID se crea automáticamente, así que no es necesario usar la función setcookie() para crearla. Y como se mantiene por defecto en la memoria del navegador, si se cierra y se vuelve a abrir éste el servidor no podrá reconocer la sesión y creará una nueva. Por ello, es recomendable incorporar un botón de "cerrar sesión" para que el usuario pueda cerrarla correctamente.

Finalmente, se debe recordar que la información almacenada en el servidor sobre la sesión seguirá existiendo hasta que pase el tiempo de expiración.

El lenguaje PHP tiene un catálogo de funciones relacionadas con el manejo de sesiones y de variables de sesión entre páginas de un mismo sitio web. No debemos olvidar que las sesiones sirven para guardar información acerca de la visita de un usuario específico, tales como nombre, apellidos, email... de forma que dicha información pueda ser recuperada en cualquier momento mientras que la sesión esté vigente.

Interesante

Funciones de PHP de manejo de sesiones

https://www.php.net/manual/es/ref.session.php

Veamos a continuación cómo es el manejo de $_SESSION

3.1 Crear sesiones

Importante

session_start()

Función que permite Iniciar una nueva sesión o reanudar la existente

En PHP, las sesiones se crean mediante la función session_start(). Si la sesión no existía, esta

función crea la sesión y le asocia un identificador de sesión único. Si la sesión ya existía, esta función permite que la página tenga acceso a la información vinculada a la sesión.

Ejemplo:

```
<?php
session_start();
?>
```

Como en el caso de las cookies, hay que tener la precaución de utilizar la función session_start() antes de empezar a escribir el contenido de la página, porque si no PHP producirá un aviso y no se creará la sesión. El motivo es que el identificador de sesión se utiliza en las cabeceras de respuesta HTTP y las cabeceras se envían antes del texto de la página. Es decir, cuando PHP encuentra una instrucción que escribe texto, cierra automáticamente la cabecera; si a continuación PHP encuentra en el programa la función session_start(), da un aviso porque ya se han enviado las cabeceras y no se crea la sesión.

☐ Interesante

Más información de la función session_start() en:
https://www.php.net/manual/es/reserved.variables.session.php

3.2 Consultar la sesión

Cuando una página ha creado una sesión o ha accedido a una sesión ya existente mediante session_start(), la página tiene acceso a la matriz $_SESSION que contiene las variables de esa sesión.

La matriz $_SESSION es una matriz asociativa en la que se pueden definir valores como en cualquier otra matriz. La diferencia es que $_SESSION es accesible desde páginas diferentes (siempre que esas páginas tengan asociada la misma sesión), manteniéndose los valores de una página a otra.

El ejemplo siguiente muestra dos páginas. La primera página guarda información en $_SESSION y la segunda la utiliza.

pagina1.php	Se ha guardado su nombre.
<?php session_start(); $_SESSION["nombre"] = "Pepe Pérez"; print " Se ha guardado su nombre. "; ?>	
pagina2.php	Su nombre es Pepe Pérez.
<?php session_start(); print " Su nombre es $_SESSION[nombre]. "; ?>	

3.3 Borrado de una variable de la sesión

El valor de las variables de $_SESSION se borran con la función **unset**().

pagina3.php	Su nombre es Pepe Pérez. No sé su nombre.

```php
<?php session_start();


if (isset($_SESSION["nombre"])) {
print "<p>Su nombre es $_SESSION[nombre].</p>";
} else {
print "<p>No sé su nombre.</p>";
}


unset($_SESSION["nombre"]);

if (isset($_SESSION["nombre"])) {
print "<p>Su nombre es $_SESSION[nombre].</p>";
} else {
print "<p>No sé su nombre.</p>";
}
?>
```

Un detalle importante que distingue a las sesiones de las cookies es que:

- En el caso de las **cookies**, cuando una página pide al navegador que cree una cookie, el valor de la cookie no está disponible en $_COOKIE en esa página. Lo estará en páginas posteriores o al refrescar la página.

- En el caso de las **sesiones**, cuando una página crea un valor en $_SESSION, ese valor está disponible en esa misma página.

> Interesante
>
> Puedes ver más información de la función unset() en el siguiente enlace:
>
> http://www.php.net/manual/es/function.unset.php
>
> Y de la matriz $_SESSION en:
>
> http://www.php.net/manual/es/reserved.variables.session.php

3.4 Destruir la sesión

El proceso para destruir una sesión es primero borrar todo el array de $_SESSION, y 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
<?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();
?>
```

> Interesante
>
> Puedes ver más información de la función session_destroy() en el siguiente enlace:
>
> http://www.php.net/manual/es/function.session-destroy.php

4. BIBLIOGRAFÍA

- Duran, C. (2013): *Cookies y sesiones*
- López, M.; Vara, JM; Verde, J.; Sánchez, D.M.; Jiménez, J.J.; Castro, V. (2012): *Desarrollo web en entorno servidor*, RA-MA, Madrid
- Vicente, J.L. (2014): *Desarrollo web en entorno servidor*, Garceta, Madrid