Tema 6 : Persistencia de la información

Ciclo Superior DAW

Asignatura: Desarrollo web en entorno servidor

Curso 21/22

Introducción

- En este capítulo veremos los siguientes conceptos:
  - Gestionar las cookies almacenadas en los navegadores de los usuarios y su información.
  - Emplear las sesiones para mantener la información de los usuarios.
  - Gestionar el inicio y cierre de las sesiones de los usuarios

Cookies

- Una **cookie** es una cantidad limitada de información que guarda el navegador en el ámbito del usuario
- Cada cookie está asociada a un sitio web determinado, de tal forma que solamente sus páginas puedan acceder a su contenido.
- Su uso más típico es el almacenamiento de las preferencias del usuario (Ejemplo: Idioma)

Cookies. Seguridad de las cookies

- **Las cookies por sí mismas no son peligrosas**, **pero algunos sitios web las emplean para rastrear la navegación de los usuarios.**
- Por ejemplo, los sitios web que añaden publicidad a las páginas pueden añadir sus propias cookies y recoger la información de aquellos sitios web visitados siempre que también contengan su publicidad.
- Estas cookies son las "**cookies de terceros**" (third-party cookies).

Cookies. Transmisión.

- **Las cookies se transmiten empleando encabezados HTTP específicos, tanto en los mensajes de petición como en los mensajes de respuesta.**

- Usamos los siguientes encabezados:
  - Para enviar: **set-cookie**
  - Para almacenar: **cookie**

Cookies. Transmisión. Enviar una cookie

- Para que un servidor almacene una cookie, envía en el mensaje HTTP de respuesta al navegador un encabezado "**Set-Cookie**" con sus datos:

```
GET: HTTP/2.0 200 OK
date: Wed, 18 Sep 2019 11:55:04 GMT
expires: -1
cache-control: private, max-age=0
content-type: text/html; charset=UTF-8
strict-transport-security: max-age=31536000
content-encoding: br
server: gws
content-length: 58938
x-xss-protection: 0
x-frame-options: SAMEORIGIN
set-cookie: 1P_JAR=2019-09-18-11; expires=Fri, 18-Oct-2019 11:55:04 GMT; path=/; domain=.google.com; SameSite=none
DV=; expires=Mon, 01-Jan-1990 00:00:00 GMT; path=/; domain=www.google.com
DV=; expires=Mon, 01-Jan-1990 00:00:00 GMT; path=/; domain=.www.google.com
DV=; expires=Mon, 01-Jan-1990 00:00:00 GMT; path=/; domain=google.com
DV=; expires=Mon, 01-Jan-1990 00:00:00 GMT; path=/; domain=.google.com
alt-svc: quic=":443"; ma=2592000; v="46,43,39"
X-Firefox-Spdy: h2
```

Cookies. Transmisión. Recuperar una cookie

- Cuando un navegador envía una petición al servidor solicitando una página, revisa las cookies y envía la información de aquellas que corresponda.
- Para esto utiliza el encabezado "**Cookie**" del mensaje de petición.

```
https://www.google.com/
Host: www.google.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:69.0) Gecko/20100101 Firefox/69.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: es-MX,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Cookie: 1P_JAR=2019-9-18-11; NID=188=WxfAUdK0B2jUibq103rqpkgQmezsRQgTPIGyyFQK-Qp5sXHHsKFZVtuKvK0mqf7657FQK8isn0TRTZ_bcLESGznQkaKfHz
Upgrade-Insecure-Requests: 1
```

Cookies. Transmisión. Recuperar una cookie

- **Los servidores pueden especificar una fecha de caducidad para las cookies.** Si no se especifica fecha de caducidad, la cookie se elimina cuando el usuario cierra el navegador (al "final de la sesión").
- A aquellas **cookies que indican una fecha de caducidad** también **se les llama "persistentes"**, pues no se eliminan al cerrar el navegador.

Cookies en PHP. Almacenamiento de información

- En PHP, **utilizamos la función "setcookie" para almacenar una cookie en el navegador del usuario.**
- Los parámetros que tiene esta función son:
  - **Nombre de la cookie** (obligatorio)
  - **Valor**
  - **Fecha de caducidad (cookie persistente)**
  - **Restricciones de acceso**

Cookies en PHP. Almacenamiento de información

- EJEMPLO: Almacenar en una cookie el idioma que prefiere el usuario para el sitio web.

```
setcookie('Lengua', 'Gallego');
```

- Esta cookie se elimina al cerrar la sesión del navegador.

Cookies en PHP. Almacenamiento de información

- EJEMPLO: Almacenar en una cookie el idioma que prefiere el usuario para el sitio web durante 1 hora

```
setcookie('Lengua', 'Gallego',  time()+3600);
```

- Esta cookie tiene un plazo de caducidad de una hora

Cookies en PHP. Almacenamiento de información

- El cuarto parámetro de la función "setcookie" sirve para aplicar restricciones a las páginas del sitio que pueden acceder a una cookie en función de la ruta.

Cookies en PHP. Almacenamiento de información

- EJEMPLO: Almacenar en una cookie la lengua que prefiere el usuario para el sitio web durante 1 hora, sólo accesible para la carpeta /Tema06/ y sus subdirectorios

```
setcookie('Lengua', 'Gallego',  time()+3600, '/Tema06/');
```

Cookies en PHP. Almacenamiento de información

- Cuando en PHP hacemos una llamada a la función "setcookie", el servidor envía un encabezado "Set-Cookie" al navegador.
- Por eso, **las llamadas a la función "setcookie" deben enviarse antes de que el navegador muestre información alguna en pantalla.**

Cookies en PHP. Recuperación de información

- Recuperar la información que almacena una cookie es muy simple.
- PHP recoge la información que el servidor recibe en los encabezados "**Cookie**" de las peticiones HTTP y acede a ella por el array "**$_COOKIE**".

Cookies en PHP. Recuperación de información

- EJEMPLO: Recuperar el valor de la lengua preferida por el usuario para una página web

```
echo 'La lengua preferida es '.$_COOKIE['Lengua'];
```

Cookies en PHP. Recuperación de información

- Siempre que una aplicación web emplee **cookies**, se debe tener bien presente que en última instancia **su disponibilidad está controlada por el cliente.**

- Por ejemplo, algunos usuarios deshabilitan por completo las cookies en el navegador

Actividad 1

Utilizando las cookies, crea un mecanismo que nos permita detectar si un usuario ya estuvo en una página web o si es nuevo

Actividad 2

Utilizando las cookies, crea un mecanismo que nos permita calcular el tiempo transcurrido desde la última vez que un usuario visitó la web

Actividad 3

Utilizando las cookies, crea un mecanismo que nos permita contar las visitas que has hecho a la página web

Cookies en PHP. Borrado de una cookie

- Las **cookies persistentes** se eliminan cuando vence su tiempo de vida.
- Las cookies en las que no se especifica el tiempo de vida son eliminadas al cerrar la sesión del navegador.

Cookies en PHP. Borrado de una cookie

- A veces necesitamos borrar una cookie del navegador.
- Esto se hace reescribiendo la misma cookie con un tiempo de vida más antiguo que el actual

Cookies en PHP. Borrado de una cookie

- EJEMPLO: Queremos borrar la cookie donde se almacenaba la lengua preferida para mostrar las páginas web

```
setcookie('Lengua', 'Gallego',  time() - 1000);
```

Actividad 4

Utilizando las cookies, crea un mecanismo que nos permita borrar todas las cookies que se han creado en los ejemplos anteriores

Cookies en PHP. Almacenamiento de arrays

- Puede ser que necesitemos guardar en una cookie los datos de un array.
- Esto se hace añadiendo tantas cookies como miembros tenga el array, de una en una, indicando sus nombres con la notación típica de los arrays.

Cookies en PHP. Almacenamiento de arrays

- EJEMPLO: Queremos guardar el nombre y apellidos de un usuario en una cookie

```
setcookie('usuario[nombre]', 'Martin');
setcookie('usuario[apellido1]', 'Garcia');
setcookie('usuario[apellido2]', 'Figueira');
```

Cookies en PHP. Almacenamiento de arrays

- EJEMPLO: Cuando recuperamos los valores vemos que se accede a la cookie como array

```php
if(isset($_COOKIE['usuario'])) {
  foreach ($_COOKIE['usuario'] as $nombre => $valor) {
    echo "$nombre:$valor <br />";
  }
}
```

Actividad 5

Utilizando las cookies, crea un mecanismo que nos permita guardar en un array

en la cookie los siguientes datos (Nombre, Apellidos, Teléfono)

Sesiones

- Una forma para guardar información particular de cada usuario es utilizar cookies.
- No obstante, **existen diversos problemas asociados a las cookies, como el número de ellas que admite el navegador, o su tamaño máximo.**
- Para solucionar estos inconvenientes, existen las **sesiones**.

Sesiones

- Una **sesión** es el **conjunto de información relativa a un usuario concreto que este almacena**.
- Esta información puede ser tan simple como el nombre del propio usuario, o más compleja, como el carrito de compra de una tienda online.
- A la información que se almacena en la sesión de un usuario se le conoce también como **cookies del lado del servidor (server side cookies).**

Sesiones. SID

- **Cada usuario distinto de un sitio web tiene su propia información de sesión.**
- Para distinguir una sesión de otra se usan los **identificadores de sesión (SID)**
- El SID de un usuario es el atributo que lo identifica y que emplea el servidor web para recuperar su información almacenada en la sesión.

Sesiones. Comunicación

- Debido a que el protocolo HTTP es un protocolo sin estado (cada petición es independiente de las anteriores), **el usuario tendrá que identificarse enviando su SID al servidor en cada petición que haga.**

Sesiones. Comunicación

- Para realizar una comunicación, el navegador del usuario tendrá que:
  - Conocer el SID del usuario.
  - Enviar el SID al servidor web junto con cada nuevo mensaje petición.
- Y cuando el servidor web reciba el SID del usuario:
  - Recuperará su información de sesión, que podrá modificar o emplear para generar el contenido del mensaje de respuesta.

Sesiones. Almacenamiento de SID

- ¿Dónde se encuentra ese SID, el identificador de la sesión, que es único para cada usuario?

- Hay dos maneras de mantener el SID entre las páginas de un sitio web que visita el usuario:

Sesiones. Almacenamiento de SID

Utilizando cookies.

- El servidor web crea una cookie en el navegador para almacenar el identificador del usuario.
- En PHP normalmente **el nombre de esa cookie es "PHPSESSID".**
- Es el método recomendado.

Sesiones. Almacenamiento de SID

Propagando el SID en un parámetro de la URL

- En PHP también suele ser "PHPSESSID".
- El SID se añade como una parte más de la URL, de la forma:

```
www.montecasteloweb.com/tienda/listado.php&PHPSESSID=34534g34ty
```

Sesiones. Almacenamiento de SID

- Ninguna de las dos maneras es perfecta.
- Sin embargo, **el mejor método y el más utilizado es la utilización de cookies**.
- Propagar el SID como parte de la URL lleva consigo mayores desventajas:
    - Que sea visible en la barra de direcciones del navegador
    - Compartir la URL con otra persona implica compartir también el identificador de sesión.

Sesiones. Almacenamiento de SID

- La manera más segura de utilizar sesiones es **almacenando los SID en cookies** y **utilizar HTTPS para encriptar la información con el SID** que se transmite entre el servidor web y el cliente.
- La buena noticia es que el proceso de manejo de sesiones en PHP está automatizado en buena medida. PHP comprueba automáticamente si existía un SID previo y tampoco es preciso programar un mecanismo “setcookie”

Sesiones. Configuración

- Es importante configurar correctamente PHP con sus correspondientes directivas antes de utilizar sesiones, en el fichero **php.ini**
- Las directivas a configurar son:
  - session.use_cookies
  - session.use_only_cookies
  - session.save_handler
  - session.save_path
  - session.name
  - session.auto_start
  - session.cookie_lifetime
  - session.cookie_secure
  - session.gc_maxlifetime
  - session.use_trans_sid

Sesiones. Configuración

Directivas de sesiones en PHP	
`session.use_cookies`	Indica si se deben usar cookies (1) o propagación en la URL (0) para almacenar el SID.
`session.use_only_cookies`	Se debe activar (1) cuando se usan cookies para almacenar los SID, para indicar que no se reconozcan los SID que se puedan pasar como parte de la URL (estos SID se pueden usar para usurpar el identificador de otro usuario).
`session.save_handler`	Se utiliza para indicar a PHP como debe almacenar los datos de la sesión del usuario. Existen 4 opciones: en ficheros (files), en memoria (mm), en una base de datos SQLite (sqlite) o utilizando funciones que debe definir el programador (user). El valor por defecto (files) funcionará sin problemas en la mayoría de los casos.
`session.save_path`	La ruta en la que se guardarán las sesiones de los usuarios, en caso de usar almacenamiento en ficheros.
`session.name`	Determina el nombre de la cookie que se utilizará para guardar el SID. Su valor por defecto es "PHPSESSID".
`session.auto_start`	Su valor por defecto es 0, y en este caso será necesario usar la función session_start para gestionar el inicio de las sesiones. Cuando un sitio web emplea sesiones con frecuencia, puede ser buena idea cambiar su valor a 1 para que PHP active de forma automática el manejo de sesiones.

Sesiones. Configuración

Directivas de sesiones en PHP	
`session.cookie_lifetime`	Establece el tiempo en segundos que se mantendrá la cookie de la sesión en el navegador del usuario, siempre que se use ese método de transmisión del SID. Su valor por defecto es 0, con el cual la cookie se mantendrá hasta que se cierre el navegador. Cuando se emplea la URL para propagar el SID, este se perderá cada vez que el usuario cierre el navegador.
`session.cookie_secure`	Cuando se activa obliga a que las cookies de sesión solamente se transmitan a través de conexiones seguras HTTPS. De esta manera, se asegura el SID frente a posibles ataques de interceptación.
`session.gc_maxlifetime`	Indica el tiempo en segundos que se debe mantener la información de la sesión, aunque no haya ninguna actividad por parte del usuario. Su valor por defecto es 1440. Pasados 24 minutos desde la última actividad por parte del usuario, los datos de su sesión pasan al recolector de basura (*garbage collector*, gc).
`session.use_trans_sid`	Activa o desactiva el soporte de PHP para el envío transparente del SID en la URL. Cuando se activa, PHP añade automáticamente el parámetro PHPSESSID a los enlaces a rutas relativas.
`url_rewriter.tags`	Cuando la directiva anterior se encuentra activada, esta indica la lista de etiquetas en las que se les añadirá de forma transparente el parámetro con el SID en las URLs.

Sesiones. Configuración

- Las directivas de la tabla anterior pueden establecerse de cualquiera de las siguientes formas:
    - En el archivo global de configuración de PHP "php.ini".
    - Dentro de los archivos "httpd.conf" y ".htaccess", en el caso de usar Apache
    - En un archivo de configuración “.user.ini", semejante a los ".htaccess" de Apache.
    - Dentro de un guión PHP, empleando una función "ini_set" o semejante.

Actividad 6

Revisa los archivos de la tabla de directivas anterior y comprueba el sitio donde se deberían cambiar las directivas

Sesiones. Inicio de sesión

- Se puede iniciar una sesión de dos formas:
  - **Si la directiva "session.auto_start" está activa**, **la sesión comenzará automáticamente** en cuanto un usuario se conecte al sitio web
  - **Si no**, será necesario **ejecutar la función "session_start"** para indicar a PHP que inicie una nueva sesión o continúe la anterior.

Sesiones. Inicio de sesión

- Para poder iniciar una sesión mediante "session_start" habrá que **hacer las llamadas a esta función antes de que la página web muestre información en el navegador**

- Además, **todas las páginas que necesiten utilizar la información almacenada en la sesión, deberán ejecutar la función "session_start".**

Sesiones. Acceso a la información

- La variable **$_SESSION** permite añadir información a la sesión del usuario, o para acceder a la información almacenada en la sesión.
- Su funcionamiento es similar al **$_POST**, **$_GET** o **$_COOKIE**, que ya vimos en anteriores ocasiones.

Sesiones. Acceso a la información

- EJEMPLO: Contar el número de veces que el usuario visita la página, guardando el contador en la sesión del usuario

```php
// Iniciamos la sesión o recuperamos la anterior sesión existente
session_start();
// Comprobamos si la variable ya existe
if (isset($_SESSION['visitas'])){
    $_SESSION['visitas']++;
}
else
{
    $_SESSION['visitas'] = 0;
}
```

Sesiones. Acceso a la información

- EJEMPLO: Guardar cada instante en que el usuario visita la página

```php
// Iniciamos la sesión o recuperamos la anterior sesión existente
session_start();
// En cada visita añadimos un valor al array "visitas"
$_SESSION['visitas'][] = mktime();
```

Sesiones. Acceso a la información

- EJEMPLO: Crear una clase Visita y guardarla en la sesión.

```php
// Iniciamos la sesión o recuperamos la anterior sesión existente
session_start();
// Creamos un nuevo objeto y lo guardamos en la sesión
$o = new Visita();
$o->setTiempo(mktime());
$_SESSION['visita'] = $o;
```

- **IMPORTANTE**: **La definición de la clase no se guarda en la variable $_SESSION, se debe cargar antes de iniciar sesión**

Sesiones. Acceso a la información

- Cuando almacenamos un objeto en la sesión, PHP lo serializa de manera automática. Y del mismo modo, cuando recuperamos un objeto de la sesión del usuario, PHP lo deserializa por nosotros.

```php
// Iniciamos la sesión o recuperamos la anterior sesión existente
session_start();
// Recuperamos el objeto de la sesión
$o = $_SESSION['visita'];
```

Sesiones. Cierre manual de la sesión

- A veces puede ser necesario cerrar la sesión de forma manual en un momento determinado. Por ejemplo, cuando se emplean sesiones para recordar la información de autenticación, conviene darle al usuario del sitio web la posibilidad de cerrar la sesión cuando lo crea conveniente.

Sesiones. Cierre manual de la sesión

- En PHP pueden emplearse **dos funciones para eliminar la información almacenada en la sesión:**
  - **session_unset**. Elimina las variables almacenadas en la sesión actual, pero no elimina la información de la sesión del dispositivo de almacenamiento usado.
  - **session_destroy**. Elimina completamente la información de la sesión del dispositivo de almacenamiento.

Actividad 7

¿Qué diferencia hay entre session_unset y session_destroy?

¿Debemos usarlas juntas?

Sesiones. Almacenamiento de la info de la sesión

- A veces puede convenir guardar en un fichero o en una base de datos la información que contiene la sesión del usuario, para poder recuperarla más adelante

- Tenemos dos funciones que facilitan el procedimiento:

Sesiones. Almacenamiento de la info de la sesión

- **session_encode**. Devuelve la información de la sesión codificada en una cadena de texto.

```
echo session_encode();
```

- **session_decode**. Recibe una cadena de texto con la información de la sesión, y con su contenido regenera el array $_ SESSION.

```
session_decode($session_data);
```

Actividad 8

Utiliza los métodos session_encode y session_decode

¿Qué nos devuelve cada una?

¿Podemos usarlas juntas?

¿Cuál es el objetivo de cada una?

Sesiones. Ataques

- Existen dos tipos distintos de ataques que se basan en el identificador de sesión:
  - **El secuestro**
  - **La fijación de la sesión**

Sesiones. Secuestro de la sesión

- Llamamos **secuestro de la sesión** a cualquier método o técnica que tenga como fin conseguir el identificador de la sesión de un usuario.
- Las formas de conseguir este objetivo son variadas, por ejemplo:
  - Interceptando el tráfico de la red
  - Realizando un ataque de fuerza bruta, probando muchos SIDs hasta conseguir uno válido.
  - Etc.

Sesiones. Secuestro de la sesión

- Para evitar en la medida de lo posible este tipo de ataques, podemos emplear las siguientes medidas:
  - **Enviar los identificadores del cliente al servidor empleando siempre cookies.**
    - De este modo, el identificador no aparece en la URL de cada una de las páginas.
    - Esto puede conseguirse empleando la siguiente directiva de configuración:

```
session.use_only_cookies = 1
```

Sesiones. Secuestro de la sesión

- Para evitar en la medida de lo posible este tipo de ataques, podemos emplear las siguientes medidas:
    - **Emplear conexiones seguras con HTTPS** para evitar la interceptación del tráfico

Sesiones. Secuestro de la sesión

- Para evitar en la medida de lo posible este tipo de ataques, podemos emplear las siguientes medidas:
  - **Limitar el tiempo de vida de la sesión**
    - **Empleando la directiva "session.cookie_lifetime" para indicar el tiempo de caducidad de una sesión.** Tras eso, la sesión pasa a considerarse inválida y deberá abrirse una nueva. También se puede con la función "session_set_cookie_params".

```
session_set_cookie_params('600'); // 10 minutos
session_start();
```

Sesiones. Secuestro de la sesión

- Para evitar en la medida de lo posible este tipo de ataques, podemos emplear las siguientes medidas:
  - **Limitar el tiempo de vida de la sesión**
    - **Empleando la directiva "session.gc_maxlifetime", que indica el tiempo de inactividad de una sesión**. Pasado ese tiempo, la información de la sesión pasa a considerarse inválida.

Sesiones. Secuestro de la sesión

- Para evitar en la medida de lo posible este tipo de ataques, podemos emplear las siguientes medidas:
  - **Regenerar el SID de la sesión de vez en cuando**
    - **Empleando la función "session_regenerate_id".** Este método es transparente para los usuarios activos, pero provoca la invalidez del anterior identificador

```
session_start();
session_regenerate_id();
```

Sesiones. Secuestro de la sesión

- Para evitar en la medida de lo posible este tipo de ataques, podemos emplear las siguientes medidas:
  - **Comprobar en cada petición los datos relacionados con el cliente**
    - Por ejemplo, el agente de usuario **($_SERVER['HTTP_USER_AGENT'])** o la dirección origen de la petición **($_SERVER ['REMOTE_ ADDR'])**
    - Cuando alguna de las dos cambia debería anularse la sesión y forzar al usuario a registrarse de nuevo para comenzar una nueva sesión.

Sesiones. Secuestro de la sesión

- Para evitar en la medida de lo posible este tipo de ataques, podemos emplear las siguientes medidas:
  - **Comprobar en cada petición los datos relacionados con el cliente**

```php
session_start();
if ($_SERVER['HTTP_USER_AGENT'] != $_SESSION['USER_AGENT_ANTERIOR'])
{
    session_destroy(); // Eliminamos la información de la sesión
}
session_regenerate_id(); // Generamos un nuevo  SID

$_SESSION['USER_AGENT_ANTERIOR'] = $_SERVER['HTTP_USER_AGENT'];
```

Sesiones. Fijación de la sesión

- **Se intenta conseguir que un usuario se loguee con un identificador de sesión que ya conocemos.**
- La forma más habitual de este ataque es hacer que el usuario emplee una URL que contenga un identificador de sesión como la siguiente:

```
www.montecasteloweb.com/tienda/listado.php&PHPSESSID=34534g34ty
```

Sesiones. Fijación de la sesión

- **Si el sitio web acepta URLs como la anterior**, con el SID incorporado, cuando el usuario entre en el enlace abrirá una nueva sesión con el identificador que figura en la URL.
- **A continuación, si el usuario inicia sesión** en el sitio web, **el atacante ya dispone de un SID** correspondiente a una sesión registrada.

Sesiones. Fijación de la sesión

- Para evitar en la medida de lo posible este tipo de ataques, debemos emplear medidas recomendadas para la fijación de la sesión
- Además, debemos **revisar el valor del parámetro "session.use_trans_sid"** en la configuración de PHP

```
session.use_trans_sid = 0;
```

- Si el valor es '1', PHP leería los identificadores de sesión pasados en un parámetro GET de la URL, lo que permitiría la creación de URLs fraudulentas

Actividad 9

¿Cuál de los dos ataques os parece más peligroso?

¿Cuál de los dos creéis que es más probable que nos pase?

Autenticación de usuarios. Contraseñas

- En muchas aplicaciones web se utilizan cuentas de usuario que tienen contraseñas para registrarse y autenticarse
- ¿Cómo almacenan estas aplicaciones web las contraseñas para autenticar a los usuarios? Pues la respuesta es que **NO LO HACEN**
- Se almacenan los "**hashes**" de las contraseñas.

Autenticación de usuarios. Hashing

- Se debe aplicar un algoritmo hash a las contraseñas antes de almacenarlos
- De esta manera, dificultamos al atacante determinar la contraseña original
- El proceso a seguir es:
    - Cuando se registre un nuevo usuario en la aplicación web deberemos **almacenar su contraseña "hasheada"**
    - Aplicando el mismo "hash", **debemos validar la contraseña introducida** por el usuario cada vez que este inicie una nueva sesión.

Autenticación de usuarios. Salt

- Un salt criptográfico es **un dato que se utiliza durante el proceso de hash para eliminar la posibilidad de que el resultado pueda buscarse a partir de una lista de pares precalculados de hash y sus entradas originales**
- Es importante emplear el mismo salt tanto en el registro de un usuario como en la comprobación de su contraseña en los sucesivos accesos a la aplicación web.

Autenticación. Mantenimiento y cierre de sesión

- Como ya vimos, **los datos de la sesión del usuario se guardan en el servidor, y el SID que identifica cada una de las sesiones se almacena normalmente en una cookie** del navegador del usuario.
- Como es información confidencial, necesitamos que se elimine y no pueda ser accesible por nadie.

Autenticación. Mantenimiento y cierre de sesión

- Veremos dos métodos que nos permitan:
  - Establecer un tiempo límite de inactividad
  - Mantener abierta la sesión del usuario, aunque este cierre su navegador.
- Estos métodos nos permitirán tener sesiones persistentes, que el usuario pueda cerrar de manera manual o que se cierren solas pasado un cierto tiempo, pero que se puedan mantener abiertas aún cuando éste cierre y abra de nuevo el navegador.

Autenticación. Mantenimiento y cierre de sesión

- Establecer un tiempo límite de inactividad
  - **Guardamos** en una variable de la sesión del usuario **el instante en** el que se produjo **la última acción**.
  - La siguiente vez que el usuario intente recuperar una página del servidor, habrá que calcular el tiempo transcurrido para ver si procede o no cerrar la sesión y eliminar su información.

Autenticación. Mantenimiento y cierre de sesión

- Establecer un tiempo límite de inactividad

```php
session_start();
$max_seg_inactivo = 600;
if(isset($_SESSION['ultima_actividad']) &&
   (time() - $_SESSION['ultima_actividad'] > $max_seg_inactivo))
{
    session_unset();
    session_destroy();
}
$_SESSION['ultima_actividad'] = time();
```

Autenticación. Mantenimiento y cierre de sesión

- Mantener abierta la sesión del usuario, aunque este cierre su navegador.
  - Para evitar que la información de la sesión del usuario se elimine cuando este cierre su navegador, podemos:
    - Cambiar el parámetro **session.cookie_lifetime** , para evitar que la sesión se destruya al cerrarlo en el navegador
    - Podemos guardar por nuestra cuenta en el navegador cierta información correspondiente al usuario que nos permita restaurar la sesión después de cerrada.
- Veamos algunos ejemplos

Autenticación. Mantenimiento y cierre de sesión

- Mantener abierta la sesión del usuario, aunque este cierre su navegador.
  - Guardar el nombre o id del usuario:

```
setcookie("login", $nombreusuario, time() + 3600);
```

  - Recuperar el nombre o id del usuario:

```
$nombreusuario = $_COOKIE['login'];
```

Autenticación. Mantenimiento y cierre de sesión

- Mantener abierta la sesión del usuario, aunque este cierre su navegador.
  - Guardar un hash calculado a partir del nombre/id del usuario y un salt.

```
$salt = '...';
setcookie("hash",  crypt($nombreusuario, $salt),  time()+3600);
```

  - Coger el hash almacenado también en la cookie, y ver si coincide con nuestros cálculos.

```
$hash = $_COOKIE['hash'];
if($hash == crypt($nombreusuario, $salt))
{
  //Iniciamos sesión para el usuario
  $_SESSION['hash'] = $nomeusuario;
}
```

Autenticación. Calcular el hash

- A continuación, veremos las opciones que tenemos en PHP para calcular el hash, pensando principalmente en su empleo con las contraseñas de los usuarios de una aplicación web.

Autenticación. Calcular el hash

Función md5(), que devuelve el hash MD5 de un texto

```
$salt = 'GKM")OI4%MGfd".g';
$contraseña = 'contraseña';
echo 'MD5:'.md5($salt.$contraseña);
//MD5:1a0d7c12f1feb1fae8c2b734638fe6fb
```

Autenticación. Calcular el hash

Función sha1(), que devuelve el hash SHA1 de un texto.

```
echo 'SHA-1:'.sha1($salt.$contrasinal);
//SHA-1:09f34f8390b858c282297cf9670c988da8581d97
```

Autenticación. Calcular el hash

Función crypt()

Aplica un algoritmo de hash sobre el texto y el salt que se le pasan y devuelve el resultado.

Puede utilizar diversos algoritmos, que veremos a continuación

Autenticación. Calcular el hash

Función crypt(), con el algoritmo DES estándar

Cuando el salt está compuesto por 2 de los siguientes caracteres:

"./0-9A- Za- z".

```
echo 'DES Estándar: '.crypt($contrasenha, '7J');
//DES Estándar: 7JC/VdPVBOpg2
```

Autenticación. Calcular el hash

Función crypt(), con el algoritmo DES extendido

Cuando el salt comienza por el carácter "_" y a continuación contiene 4 caracteres para indicar el número de iteraciones y otros 4 para el salt propiamente dicho. Estos ocho caracteres deben ser del conjunto "./0-9A- Za- z".

```
echo 'DES Extendido: '.crypt($contrasenha, '_AA..5T0K');
//DES Extendido: _AA..5T0KE0q8CWN4tpg
```

Autenticación. Calcular el hash

Función crypt(), con el algoritmo MD5

Cuando el salt contiene doce caracteres comenzando por "1".

```
echo 'MD5:'.crypt($contrasenha, '$1$6YK/09.sd2');
//MD5: $1$6YK/09.s$SwmnpM9GPlQFiQllLUu920
```

Autenticación. Calcular el hash

Función crypt(), con el algoritmo Blowfish

Cuando el salt comienza por "$2la$", "$2x$" o "$2y$". A continuación debe figurar el número de iteraciones, entre 4 y 31, un signo "$", y 22 caracteres del alfabeto "./0-9A- Za- z". En las versiones de PHP 5.3.7 y superiores, se debería escoger "$2 y$" como cadena inicial.

```
echo 'Blowfish:'.crypt($contrasenha, '$2y$08$j/2Do/3dGF..Ep9s//dA4B');
//Blowfish: $2y$08$j/2Do/3dGF..Ep9s//dA4.BvtbJWNkQX0xz4zmiHpZtdXnPFUc61W
```

Autenticación. Calcular el hash

Función crypt(), con el algoritmo SHA-256

Cuando el salt comienza por "5" y contiene otros 16 caracteres.

```
echo 'SHA256:'.crypt($contrasenha, '$5$GKM")OI4%MGfd".g');
// SHA256: $5$GKM")OI4%MGfd".g$izVoWp0CrIwSFVr6tuHUIQkosFWGBluPPDv.q.syEP8
```

Autenticación. Calcular el hash

Función crypt(), con el algoritmo SHA-512

Cuando el salt comienza por "6" y contiene otros 16 caracteres.

```
echo 'SHA512:'.crypt($contrasenha, '$6$rounds=5000$GKM")OI4%MGfd".g');
// SHA512: $6$rounds=5000$GKM")OI4%MGfd".g$Zkx4X7mVloVOdtY/fpEemyC7NRw
// /yT6ocgL932f5fgZwCAmkXTAFitf2En2NBt5qit6UPJHuglPcpJJZynP7C0
```

Autenticación. Calcular el hash

Emplear la extensión Password hashing

- Se basa en dos funciones:
  - **password_hash().** Crea un hash a partir de un texto y una constante que indique el algoritmo a emplear.
  - **password_verify().** Recibe como parámetros la contraseña y el hash almacenado, e indica si la contraseña es correcta (true) o no (false).

Autenticación. Calcular el hash

Emplear la extensión Password hashing

- **password_hash().** Hay dos opciones:
  - **PASSWORD_BCRYPT**: Usa el algoritmo "bcrypt", una variante de Blowfish.
  - **PASSWORD_DEFAULT**: Usa el algoritmo que esté considerado como más seguro y puede variar con cada versión de PHP.

Autenticación. Calcular el hash

Emplear la extensión Password hashing

- **password_hash().**

```php
$hash = password_hash($contrasenha, PASSWORD_DEFAULT);
echo 'Password hash: '.$hash."<br />";
// Password hash:
//$2y$10$WG94Mxlvd.oWK5D0MDh4DekuIqMnbwttimlCJfK3iNjVNNrqcRc46
```

Autenticación. Calcular el hash

Emplear la extensión Password hashing

- **password_verify()**

```php
if (password_verify($contrasenha, $hash))
{
    echo 'La contraseña es válida!';
}
else
{
    echo 'La contraseña es inválida!';
}
```

Actividad 10

Realiza un ejemplo utilizando las funciones password_hash y password_verify

KEEP CALM

IT'S KAHOOT TIME

Tema 6 : Persistencia de la información

Ciclo Superior DAW

Asignatura: Desarrollo web en entorno servidor

Curso 21/22