

UT5 DESARROLLO DE APLICACIONES WEB UTILIZANDO CÓDIGO EMBEBIDO

Índice

- - Autenticación y control de acceso
 - Mecanismos de autenticación
 - Cookies
 - Gestión de sesiones

Autenticación y control de acceso

Es importante verificar la identidad de los dos extremos de una comunicación.

Se debería utilizar el protocolo HTTPS.

En la mayoría de las aplicaciones web existe un mecanismo de control de acceso que obligue al usuario a identificarse.

Mecanismos de autenticación

El protocolo HTTP ofrece un método sencillo para autenticar a los usuarios:

- El servidor web debe proveer algún método para **definir los usuarios** que se utilizarán y cómo se pueden autenticar.
- Cuando **un usuario no autenticado intenta acceder** a un recurso restringido, **el servidor web responde con un error** de "Acceso no autorizado" (código 401).
- El **navegador** recibe el error y **abre una ventana** para solicitar al usuario que se autentique mediante **su nombre y contraseña**.
- La información de autenticación del usuario **se envía al servidor**, que **la verifica** y decide si permite o no el acceso al recurso solicitado. **Esta información se mantiene en el navegador** para utilizarse en posteriores peticiones a ese servidor.

En Apache existe la utilidad **htpasswd**

- Fichero con usuarios y contraseñas
- Crearlo en un lugar no accesible por los usuarios del servidor web

Pero ¿cómo le indicamos a Apache qué recursos tienen acceso restringido? Respuesta: Fichero .htaccess

- Además tendrás que asegurarte de que en la configuración de Apache se utiliza **la directiva AllowOverride** para que se aplique correctamente la configuración que figura en el **fichero .htaccess**.

Desde PHP puedes acceder a la información de autenticación HTTP que ha introducido el usuario utilizando el array superglobal **\$_SERVER**

\$_SERVER['PHP_AUTH_USER'] Nombre de usuario que se ha introducido.

\$_SERVER['PHP_AUTH_PW'] Contraseña introducida.

\$_SERVER['AUTH_TYPE'] Método HTTP usado para autenticar. Puede ser Basic o Diges.

En PHP puedes usar **la función header** para forzar a que el servidor envíe un error de "Acceso no autorizado" (código 401). De esta forma no es necesario utilizar ficheros .htaccess para indicarle a Apache qué recursos están restringidos.

En su lugar, puedes añadir las siguientes líneas en tus páginas PHP:

```
if (!isset($_SERVER['PHP_AUTH_USER'])) {
    header('WWW-Authenticate: Basic Realm="Contenido restringido"');
    header('HTTP/1.0 401 Unauthorized');
    echo "Usuario no reconocido!";
    exit();
}
```

Te habrás dado cuenta en el ejercicio anterior que ahora se solicitan credenciales HTTP, pero el servidor no verifica la información. Debemos ser nosotros los que lo comprobemos.

```
if ($_SERVER['PHP_AUTH_USER'] != 'usuario' || $_SERVER['PHP_AUTH_PW'] != 'contraseña')
{
    header('WWW-Authenticate: Basic Realm="Contenido restringido"');
    header('HTTP/1.0 401 Unauthorized');
    echo "Usuario no reconocido!";
    exit();
}
```

Pregunta:

¿Creéis que esto es correcto? ¿Veis algún inconveniente? ¿Se os ocurre alguna alternativa?

Una solución mejor es utilizar **un almacenamiento externo** para los nombres de usuario y sus contraseñas. Para esto podrías emplear un **fichero de texto**, o mejor aún, **una base de datos**. La información de autenticación podrá estar aislada en su propia base de datos, o compartir espacio de almacenamiento con los datos que utilice tu aplicación web.

Pregunta


¿Cómo almacenaríais la contraseña?

MD5 es un método para generar un resumen de un texto o un documento, de tal forma que a partir del resumen obtenido no es posible recuperar el texto original, ni hallar otro texto a partir del cual se obtenga el mismo resumen. Se llama hash al resumen obtenido al aplicar una función hash. Una de las funciones hash más extendidas es MD5, que genera 128 bits como resumen (normalmente se representa mediante una cadena de texto de 28 caracteres o mediante 32 dígitos hexadecimales).

En PHP puedes usar la función md5 para calcular el hash MD5 de una cadena de texto

```
$str = 'contraseña';
if (md5($str) == '4c882dcb24bcb1bc225391a602feca7c') {
```

```
    echo "Contraseña correcta";
}
```

 Hoja05_Sesiones_01

Cookies



Pero, ¿qué es una **cookie**?

Una cookie es un fichero de texto que un sitio web guarda en el entorno del usuario del navegador.

Su uso más típico es el almacenamiento de las preferencias del usuario (por ejemplo, el idioma en que se deben mostrar las páginas), para que no tenga que volver a indicarlo la próxima vez que visite el sitio.

El objetivo esencial de una cookie es identificar nuestro usuario al almacenar y alojar el historial de actividad de dicho sitio web, esto permite que el sitio web identifique los usos habituales en esa página y en base a ello ofrecer las mejores alternativas de contenido al usuario.

Recordemos que a nivel web existen algunos tipos de cookie como son:

- **Zombie cookies** las cuales tienen la capacidad de regenerarse una vez han sido borradas del sistema ya que se alojan en el sistema mas no directamente en el navegador.
- **Secure cookies** las cuales almacenan la información usando métodos de cifrado con el fin de impedir que los datos guardados estén abiertos a ataques maliciosos y este tipo de cookies solo es usada en conexiones HTTPS.
- **Session cookies** las cuales como su nombre lo dice, son las cookies de cada sesión iniciada en el navegador y estas serán borradas al cerrar el navegador.
- **Persistent cookies** las cuales son las más frecuentes y tienen como misión rastrear todos los procesos del usuario al almacenar información sobre todo lo que hace en el sitio web tomando como base un período de tiempo determinado, estas cookies persistentes son borradas cuando se eliminan los datos del navegador usado.

En PHP, para almacenar una cookie en el navegador del usuario, podemos utilizar la función **setcookie**

Pregunta

¿Qué hará la siguiente instrucción?

```
setcookie("nombre_usuario", $_SERVER['PHP_AUTH_USER'], time()+3600);
```

¿Os parece que es aconsejable guardar esta información en cookies?

El proceso de recuperación de la información que almacena una cookie es muy simple. Cuando accedes a un sitio web, el navegador le envía de forma automática todo el contenido de las cookies que almacene relativas a ese sitio en concreto.

Desde PHP puedes acceder a esta información por medio del array **\$_COOKIE**



Hoja05_Sesiones_02

Gestión de sesiones

El término sesión hace referencia al conjunto de información relativa a un usuario concreto.

Ejemplos:

- Nombre del usuario
- Artículos de la lista de la compra de una tienda online

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)**.

Un **SID** es un atributo que se asigna a cada uno de los visitantes de un sitio web y lo identifica.

Pregunta: ¿Dónde se almacena ese SID?

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

- Utilizando cookies, tal y como vimos
- Propagando el SID en un parámetro de la URL. El SID se añade como una parte más de la URL, de la forma:

<http://www.misitioweb.com/tienda/listado.php&PHPSESSID=34534fg4ffg34ty>

En PHP el manejo de sesiones está automatizado en gran medida.

- Cuando un usuario visita un sitio web, no es necesario programar un procedimiento para ver si existe un SID previo y cargar los datos asociados con el mismo. Tampoco tienes que utilizar la función `setcookie` si quieres almacenar los SID en cookies, o ir pasando el SID entre las páginas web de tu sitio si te decides por propagarlo. Todo esto PHP lo hace automáticamente.
- Por defecto, PHP incluye soporte de sesiones incorporado.

Sin embargo, antes de utilizar sesiones en tu sitio web, debes configurar correctamente PHP utilizando las siguientes directivas en el **fichero php.ini** según corresponda.

Directiva	Significado
session.use_cookies	Indica si se deben usar cookies (1) o propagación en la URL (0) para almacenar el SID

Directiva	Significado
session.use_only_cookies	Se debe activar (1) cuando utilizas cookies para almacenar los SID, y además no quieres que se reconozcan los SID que se puedan pasar como parte de la URL (este método se puede usar para usurpar el identificador de otro usuario)
session.save_handler	Se utiliza para indicar a PHP cómo debe almacenar los datos de la sesión del usuario. Existen cuatro opciones: en ficheros (files), en memoria (mm), en una base de datos SQLite (sqlite) o utilizando para ello funciones que debe definir el programador (user). El valor por defecto (files) funcionará sin problemas en la mayoría de los casos
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 deberás usar la función session_start para gestionar el inicio de las sesiones. Si usas sesiones en el sitio web, puede ser buena idea cambiar su valor a 1 para que PHP active de forma automática el manejo de sesiones
session.cookie_lifetime	Si utilizas la URL para propagar el SID, éste se perderá cuando cierres tu navegador. Sin embargo, si utilizas cookies, el SID se mantendrá mientras no se destruya la cookie. En su valor por defecto (0), las cookies se destruyen cuando se cierra el navegador. Si quieres que se mantenga el SID durante más tiempo, debes indicar en esta directiva ese tiempo en segundos
session.gc_maxlifetime	Indica el tiempo en segundos que se debe mantener activa la sesión, aunque no haya ninguna actividad por parte del usuario. Su valor por defecto es 1440. Es decir, pasados 24 minutos desde la última actividad por parte del usuario, se cierra su sesión automáticamente


El inicio de una sesión puede tener lugar de dos formas:

- Si has activado la directiva **session.auto_start** en la configuración de PHP, la sesión comenzará automáticamente en cuanto un usuario se conecte a tu sitio web.
- Si no se utiliza el inicio automático de sesiones, habrá que ejecutar la función **session_start** para indicar a PHP que inicie una nueva sesión o reanude la anterior. Esta función devuelve false en caso de no poder iniciar o restaurar la sesión.
-

Mientras la sesión permanece abierta, puedes utilizar la variable superglobal **\$_SESSION** para añadir información a la sesión del usuario, o para acceder a la información almacenada en la sesión.

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. Sería similar a hacer `$_SESSION = array();`
- **session_destroy**. Elimina completamente la información de la sesión del dispositivo de almacenamiento.

 Hoja05_Sesiones_03