

UT5. Técnicas para mantener el estado y autenticación de usuarios

DAW - Desarrollo Web Entorno Servidor
Fernando Galindo

1. Cookies
2. Sesiones
3. Mecanismos de autenticación y seguridad
4. Herramientas de depuración
5. Redirecciones y códigos de estado

Las aplicaciones que hemos desarrollado carecen de estado y no somos capaces de mantener información o almacenar información de la ejecución.

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

Una cookie es útil para almacenar información del usuario en el navegador, como preferencias o idioma, para que no se tengan que volver a crear al visitar la misma página.

Las cookies se crean a petición del servidor Web, que crea el archivo en el navegador.

Los servidores web sólo pueden acceder a cookies establecidas a su propio dominio. Este dominio es establecido por el navegador cuando el servidor crea una nueva cookie, y sólo puede ser un dominio o subdominio del servidor.

Las cookies tienen una fecha de expiración, esto permite eliminar cookies antiguas.

Si la fecha de expiración no existe, será considerada una cookie de sesión y que se eliminará cuando se cierre la ventana o pestaña del navegador

La función `setcookie()` nos permite establecer una cookie.

```
setcookie(  
    string $name,  
    string $value = "",  
    int $expires = 0  
): bool  
  
setcookie("CookiePrueba", "Texto que quiero guardar" ,  
time()+3600); /* expira en 1 hora */  
  
echo $_COOKIE["CookiePrueba"];
```

Si queremos borrar una cookie podemos establecer la cookie con una fecha de expiración anterior

Algunos navegadores pueden limitar el uso de las cookies y no permitir la creación, incluso el usuario puede borrarlo.

```
setcookie('_gid','fernando',time()+3600,'/dwes/cookie','localhost');
```

Las cookies aunque tienen sus limitaciones y la información que se puede almacenar en ellas, sobre todo si es de carácter sensible

Imaginemos que queremos almacenar el carrito de la compra de un determinado usuario y esta información se guarde en el servidor (por seguridad)

El servidor conoce al usuario y le muestra su carrito con los artículos que ha añadido, esto se realiza a través de las **sesiones**

Una sesión identifica a un usuario navegador dentro del sitio Web

Esto se consigue con una cookie especial que identifica a dicho usuario

En PHP esta variable tiene el nombre de PHPSESSID

Name	Value
PHPSESSID	n76h8c9vgctu4rtcqpcbkqhvvn

Otros lenguajes o páginas utilizan las cookies SID, _GID, HSID, ... para almacenar esta información

La variable identifica al usuario y por tanto podemos recuperar información guardada de dicho usuario


```
session_start();  
  
echo 'Iniciamos la sesión';  
  
$_SESSION['alumno'] =  
    'Fernando';  
  
$_SESSION['modulo'] =  
    'DWES';  
  
$_SESSION['nota'] = '5.0';  
  
$_SESSION['instante'] =  
    time();
```

```
session_start();  
  
echo 'Recuperamos la  
sesión';  
  
echo $_SESSION['alumno'];
```

`session_unset()` - Elimina las variables almacenadas en la sesión actual, pero no elimina la información de la sesión del dispositivo de almacenamiento usado.

Para liberar una variable debemos utilizar `unset($_SESSION['var'])`

`session_destroy()` - Elimina completamente la información de la sesión del dispositivo de almacenamiento.

```
session_start(); //Iniciamos una sesión
```

```
$_SESSION=array(); //Array nuevo sin elementos
```

La cookie de sesión si es interceptada puede suplantar a un usuario por lo que es necesario utilizar más mecanismos de seguridad como cifrado y conexiones seguras

Las cookies y otras tecnologías que se usan para autenticar a los usuarios permiten asegurar que solo el propietario de una cuenta puede acceder a ella. Por ejemplo, las cookies "SID" y "HSID" contienen registros cifrados y firmados de forma digital del ID de cuenta de Google de un usuario y del momento de inicio de sesión más reciente. La combinación de estas cookies permite a Google bloquear muchos tipos de ataques, como, por ejemplo, intentos de robo del contenido de los formularios que se envían en los servicios de Google.

<https://policies.google.com/technologies/cookies?hl=es>

3. Mecanismos de autenticación y seguridad

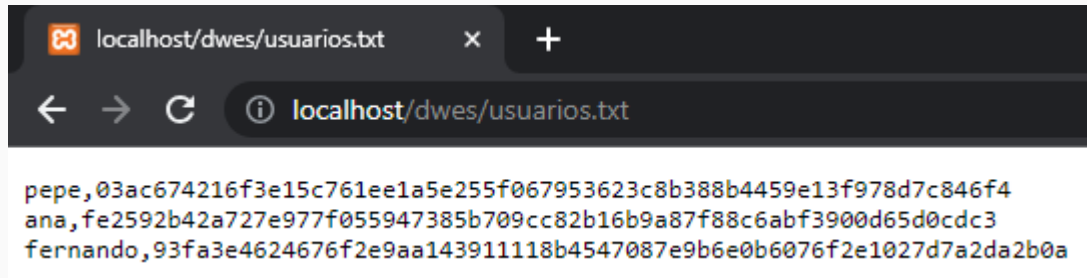


Hemos utilizado formularios para autenticar usuarios o realizar sesiones

La información la hemos guardado bien dentro del código o bien en ficheros dentro del servidor

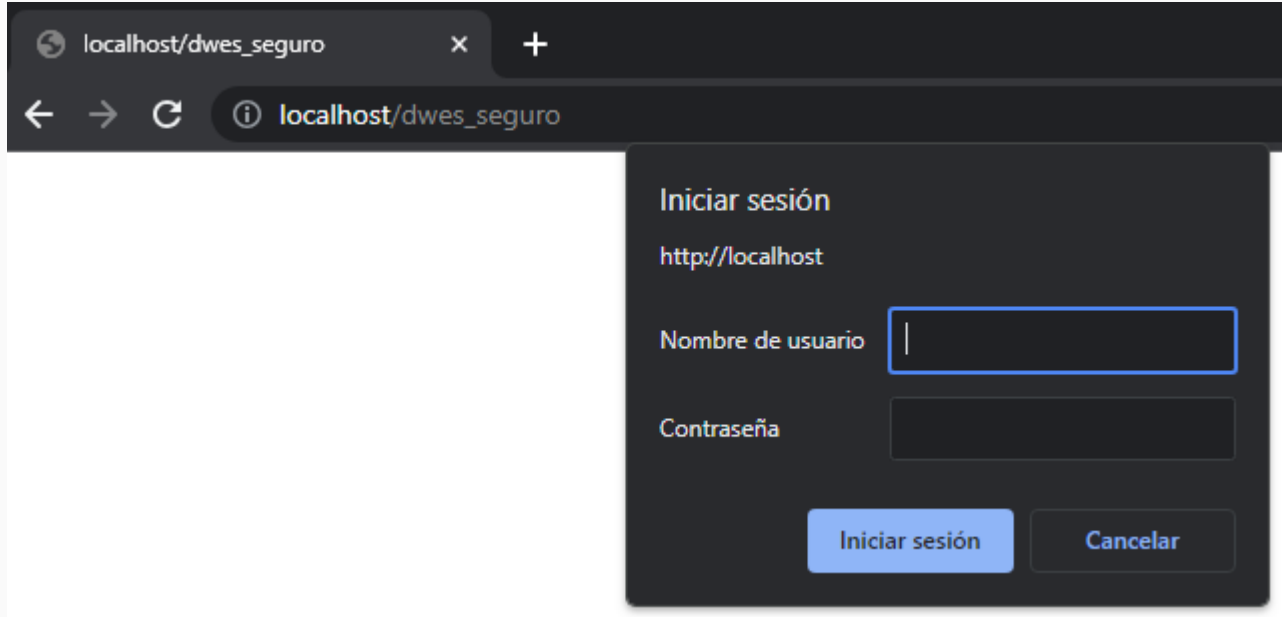
```
$fp=fopen("usuarios.txt", "a");  
fwrite($fp, $user.", ".hash("sha256", $pass));
```

Sin embargo, estos ficheros están accesibles y visibles por cualquier usuario que conozca el nombre del fichero donde guardamos estos usuarios



3. Mecanismos de autenticación y seguridad

Resulta conveniente que determinadas rutas o ficheros no estén accesibles y sólo determinados usuarios puedan acceder a ellos, esto se consigue con los ficheros `.htaccess` y `.htpasswd`



The image shows a web browser window with the address bar displaying `localhost/dwes_seguro`. A login dialog box is overlaid on the page. The dialog has the title "Iniciar sesión" and the URL `http://localhost`. It contains two input fields: "Nombre de usuario" (Username) and "Contraseña" (Password). At the bottom of the dialog are two buttons: "Iniciar sesión" (Login) and "Cancelar" (Cancel).

3. Mecanismos de autenticación y seguridad



Fichero `.htaccess`

Este fichero se crea en el directorio que queremos securizar, la estructura es la siguiente

```
AuthName "Este es mi sitio secreto!!.. Identificate."
```

```
AuthType Basic
```

```
AuthUserFile .htpasswd
```

```
require valid-user
```

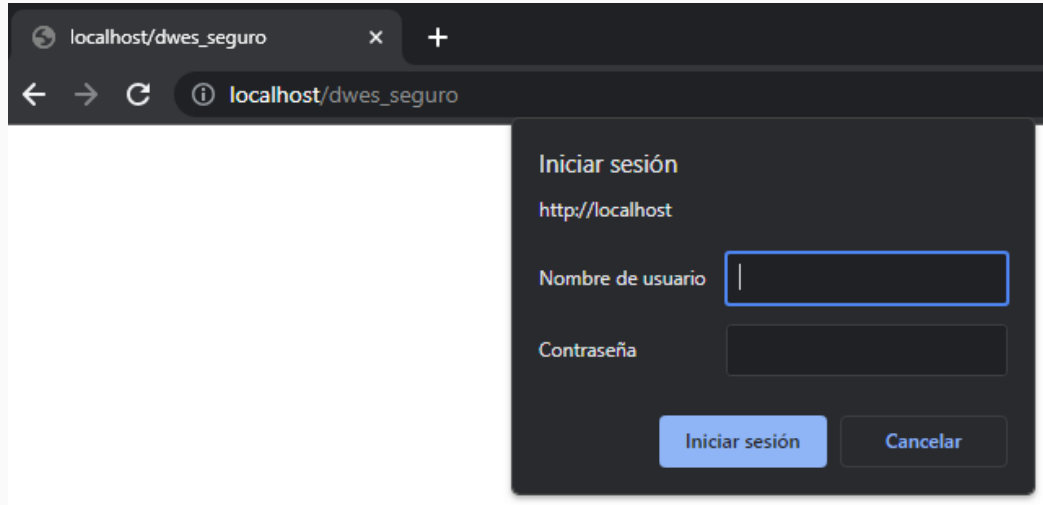
Resulta conveniente que el fichero `.htpasswd` no esté en una ruta navegable por el servidor, así que lo crearemos en otra ruta

```
AuthUserFile C:\xampp\usuarios\.htpasswd
```

3. Mecanismos de autenticación y seguridad

Al realizar llamadas dentro de la ruta donde hemos colocado el fichero
`.htaccess`

Nos pedirá un usuario y contraseña para poder navegar, estos usuarios y contraseñas están dentro del fichero `.htpasswd`



3. Mecanismos de autenticación y seguridad



El fichero `.htpasswd` se genera con el programa `htpasswd` que está en la instalación de apache

```
C:\Users\admin>\xampp\apache\bin\htpasswd.exe -c -s .htpasswd fernando
New password: ****
Re-type new password: ****
Adding password for user fernando
```

```
C:\Users\admin>C:\xampp\apache\bin\htpasswd.exe
Usage:
    htpasswd [-cimBdpsDv] [-C cost] passwordfile username
    htpasswd -b[cmBdpsDv] [-C cost] passwordfile username password

    htpasswd -n[imBdps] [-C cost] username
    htpasswd -nb[mBdps] [-C cost] username password

-c Create a new file.
-n Don't update file; display results on stdout.
-b Use the password from the command line rather than prompting for it.
-i Read password from stdin without verification (for script usage).
-m Force MD5 encryption of the password (default).
-B Force bcrypt encryption of the password (very secure).
-C Set the computing time used for the bcrypt algorithm
  (higher is more secure but slower, default: 5, valid: 4 to 17).
-d Force CRYPT encryption of the password (8 chars max, insecure).
-s Force SHA encryption of the password (insecure).
-p Do not encrypt the password (plaintext, insecure).
-D Delete the specified user.
-v Verify password for the specified user.
On other systems than Windows and NetWare the '-p' flag will probably not work.
The SHA algorithm does not use a salt and is less secure than the MD5 algorithm.

C:\Users\admin>
```


3. Mecanismos de autenticación y seguridad

Podemos acceder a las credenciales que ha introducido el usuario a través de las variables globales

<code>\$_SERVER['PHP_AUTH_USER']</code>	Nombre de usuario que se ha introducido.
<code>\$_SERVER['PHP_AUTH_PW']</code>	Contraseña introducida.
<code>\$_SERVER['AUTH_TYPE']</code>	Método HTTP usado para autenticar. Puede ser Basic o Digest.

4. Herramientas de depuración



Conocemos herramientas con las podemos ver el estado de las variables en un determinado instante

```
echo, var_dump, print_r
```

Trabajar con estas funciones puede resultar tedioso a la hora de realizar investigaciones más exhaustivas en nuestros códigos

Podemos apoyarnos en herramientas externas de depuración como xdebug para ayudarnos en el análisis

<https://xdebug.org/docs/install#windows>

5. Redirecciones y códigos de estado

Las llamadas a páginas Web tienen un código de estado que indica si una petición se ha realizado de forma satisfactoria

- Respuestas informativas (100–199)
- Respuestas satisfactorias (200–299)
- Redirecciones (300–399)
- Errores de los clientes (400–499)
- Errores de los servidores (500–599)

Estado	Método	Dominio
302	GET	localhost
200	GET	localhost
404	GET	localhost

<https://developer.mozilla.org/es/docs/Web/HTTP/Status>

5. Redirecciones y códigos de estado



La función header nos permite realizar redirecciones (3xx) que quedan registradas, de igual forma que los accesos no autorizados a determinadas páginas queden registrados con los códigos de error

Podemos incluir información de la respuesta en la función header

```
header('HTTP/1.0 401 Unauthorized');
```

O bien utilizar la siguiente función de forma más simple

```
http_response_code('401');
```

5. Redirecciones y códigos de estado



```
<?php
if (!isset($_SERVER['PHP_AUTH_USER'])) {
    header('WWW-Authenticate: Basic Realm="Contenido restringido"');
    header('HTTP/1.0 401 Unauthorized');
    echo "Usuario No Valido";
    exit;
}
?>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<title>Ejercicio: Función header para autenticación HTTP</title>
<link href="dwes.css" rel="stylesheet" type="text/css">
</head>
<body>
<?php
echo "Nombre de usuario: ".$_SERVER['PHP_AUTH_USER']."<br />";
echo "Contraseña: ".$_SERVER['PHP_AUTH_PW']."<br />";
?>
</body>
</html>
```