

## **UNIDAD DIDÁCTICA 9:**

### **Sesiones, seguridad y autenticación**

**Módulo profesional:**  
**Desarrollo Web en Entorno Servidor**

## Contenido

RESUMEN INTRODUCTORIO .....	3
INTRODUCCIÓN .....	3
CASO INTRODUCTORIO.....	3
1. SESIONES .....	4
1.1. Superglobal.....	7
1.2. Sesiones.....	8
1.3. Destruyendo sesiones.....	10
1.4. PASO 1: Objeto Seguridad .....	12
1.5. PASO 2: El formulario de registro.....	13
1.6. PASO 3: La codificación de la contraseña .....	14
1.7. PASO 4: El formulario de acceso .....	16
1.8. PASO 5: Activación de la sesión .....	17
1.9. PASO 6: Proteger una página .....	18
1.10. PASO 7: LogOut.....	19
2. COOKIES.....	20
2.1. Creando y recuperando cookies .....	22
2.2. Aplicación de las cookies al login.....	24
Resumen final: .....	26

## RESUMEN INTRODUCTORIO

En esta unidad aprenderemos los conceptos necesarios para poder manejar sesiones y cookies, así como las diferencias entre ambos conceptos.

Una vez comprendidos los conceptos básicos podremos aplicarlos en la seguridad y control de acceso básico para nuestras aplicaciones.

## INTRODUCCIÓN

Cualquier aplicación moderna tiene una zona de acceso restringido así como un registro e incluso una diferenciación de uso por roles de usuario. No es nada complejo implementarlo y con PHP es muy sencillo con la combinación y uso de sesiones y cookies.

Las sesiones nos permiten controlar desde la parte de servidor información referente para una petición de cliente. Las cookies nos permiten controlar información desde la parte de navegador o de cliente. La combinación de ambas propiedades nos permite realizar un control de acceso y seguridad completo para una determinada aplicación.

El uso de sesiones y cookies no será complejo ya que pertenecen al entorno de superglobals, es decir variables que no hace falta crear ni referencias y que son arrays asociativos que nos permiten almacenar y gestionar información, de igual forma a como realizábamos con GET o POST.

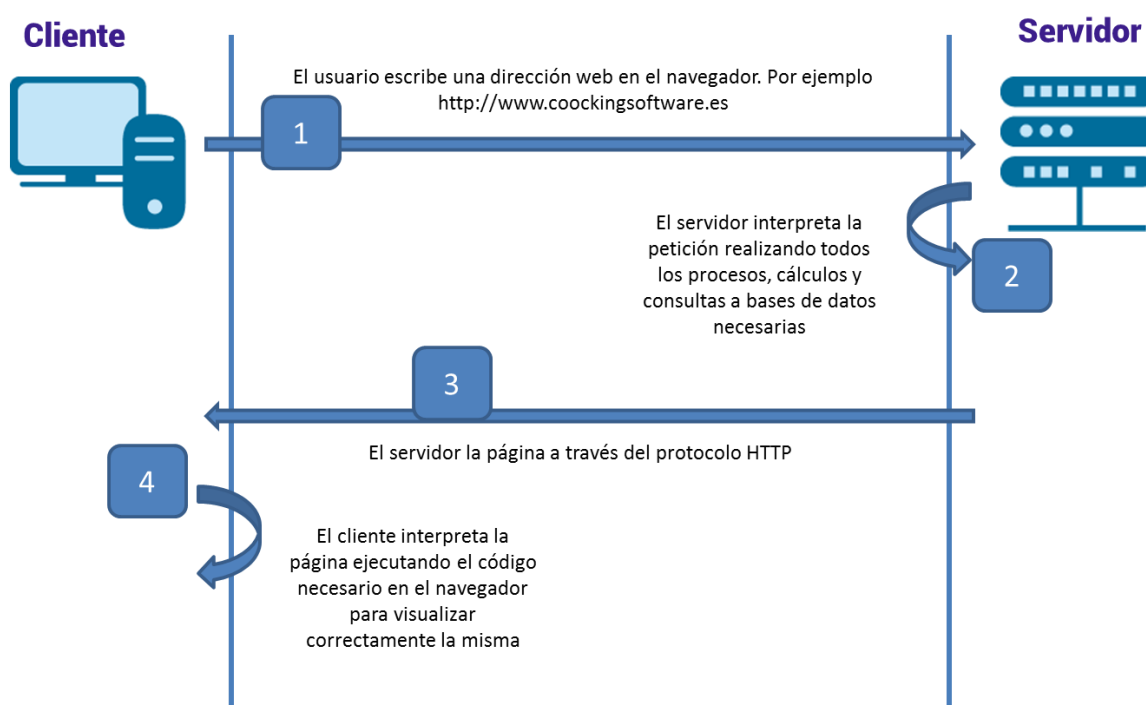
## CASO INTRODUCTORIO

Queremos realizar introducir un control de acceso y una gestión de perfil a cualquier aplicación realizada.

## 1. SESIONES

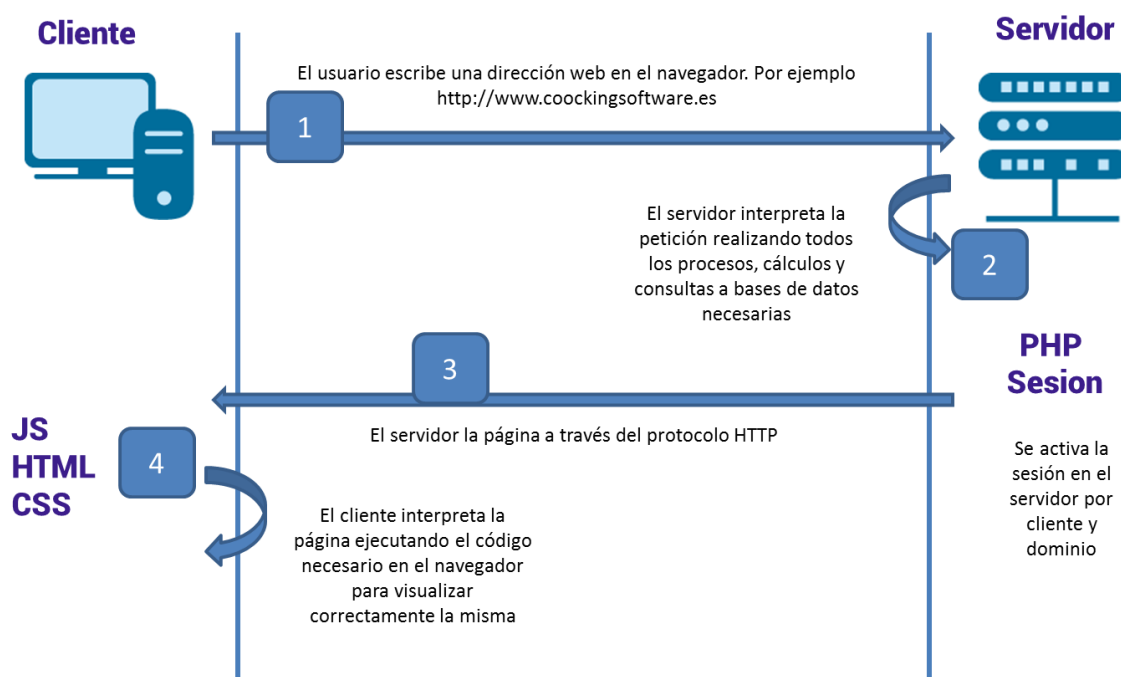
Las sesiones, al igual que las cookies que veremos en el siguiente apartado, se encuentran dentro de lo que PHP denomina superglobals. En este epígrafe las veremos y estudiaremos su significado.

Es importante tener en cuenta la diferencia entre una sesión y las variables que intervienen, y una cookie y su uso.



**Imagen 1: Secuencia de petición HTTP.** Fuente de la imagen: propia.

Recordamos y retomamos el esquema que en las primeras Unidades utilizábamos para describir una secuencia de petición por parte de un cliente de una página HTTP.

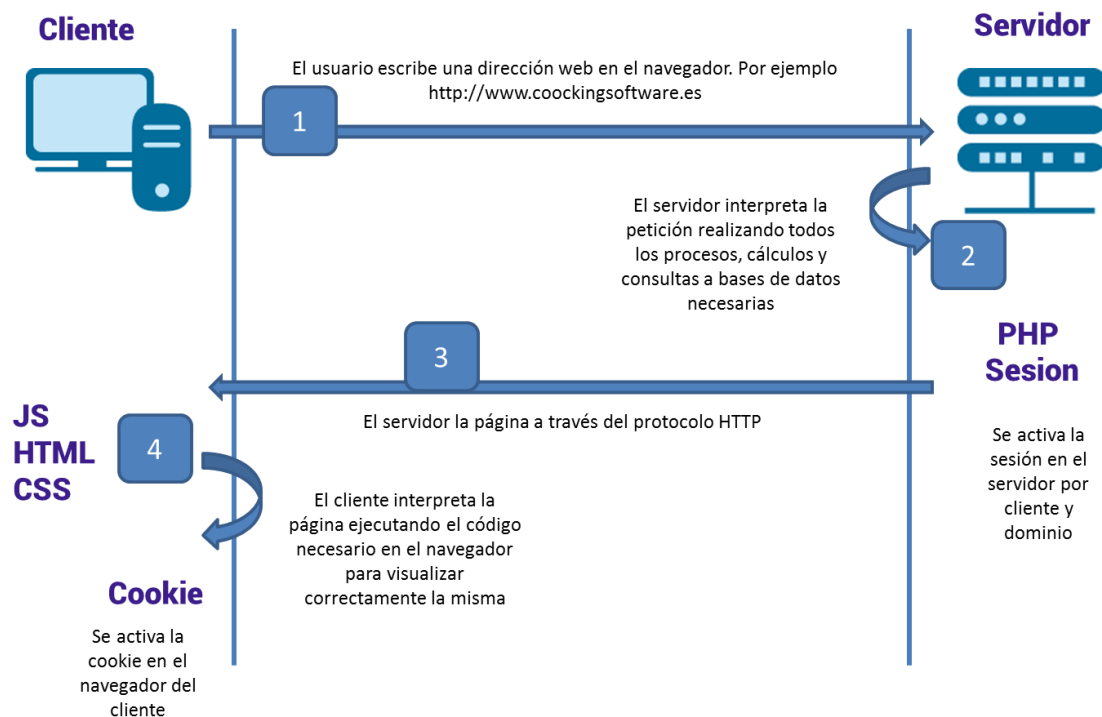


**Imagen 2: Activación de la sesión.** Fuente de la imagen: propia.

Como vemos en el anterior ejemplo, una vez que el cliente realiza la petición y esta alcanza el servidor, se activa una nueva sesión EN EL SERVIDOR, por cliente y por dominio. Es importante, y por eso lo remarcamos, que la sesión se controla por y en el SERVIDOR. Esta sesión estará por lo tanto activa por dos motivos:

- Siempre que el cliente mantenga el navegador abierto, independientemente de que todas las pestañas o conexiones estén cerradas contra ese servidor. Es decir, imaginemos que yo me tengo Chrome abierto, me conecto a la página web [www.php.net](http://www.php.net), abro otras pestañas a otras URL, y cierro la pestaña de [www.php.net](http://www.php.net). La sesión seguiría estando activa y abierta.
- La sesión se cerrará en el anterior caso cuando el navegador se cierre.
- El servidor cierra la sesión, bien porque haya pasado un tiempo programado, bien porque el cliente envíe esa acción (el típico *logout*), bien porque el servidor decida por funcionamiento de la aplicación cerrar la sesión.

El caso de las cookies es diferente:



**Imagen 3: Cookies vs sesiones.** Fuente de la imagen: propia.

Como observamos, las cookies se crean y se mantienen en el cliente, es más, la cookie se crea en un navegador en concreto, esto es, si nosotros tenemos abiertos dos navegadores diferentes, Chrome y Firefox, por ejemplo, se pueden perfectamente tener dos cookies diferentes.

Así pues:

- Las sesiones están relacionadas con el servidor, las cookies con el cliente.
- Las sesiones se cierran al cerrar el navegador, las cookies no se destruyen al cerrar el navegador.
- Las sesiones son temporales (mientras dura la sesión o bien el navegador abierto), las cookies son permanentes hasta que son destruidas.

## BIBLIOGRAFÍA RECOMENDADA



**Libro recomendado:** *Hacking with PHP* (Hudson, 2015).

En el capítulo 10 encontramos la referencia a ejemplos y documentación sobre el manejo de sesiones y cookies.

<http://www.hackingwithphp.com/10/0/0/cookies-and-sessions>

### 1.1. Superglobal

Algunas variables predefinidas en PHP son "superglobales", lo que significa que están disponibles en todos los ámbitos a lo largo del script. No es necesario emplear global \$variable; para acceder a ellas dentro de las funciones o métodos.

Según la documentación de php.net tenemos las siguientes superglobals:

#### SUPERGLOBAL

```
$GLOBALS  
$_SERVER  
$_GET  
$_POST  
$_FILES  
$_COOKIE  
$_SESSION  
$_REQUEST  
$_ENV
```

Vemos en este listado algunas de esas variables, como \$\_POST y \$\_GET, y como veremos el uso y acceso de las mismas será idéntico.

## 1.2. Sesiones

Las sesiones son una forma sencilla de almacenar datos para usuarios de manera individual usando un ID de sesión único. Esto se puede usar para hacer persistente la información de estado entre peticiones de páginas. Los ID de sesiones normalmente son enviados al navegador mediante cookies de sesión, y el ID se usa para recuperar los datos de sesión existente. La ausencia de un ID o una cookie de sesión permite saber a PHP para crear una nueva sesión y generar un nuevo ID de sesión.

Veamos cómo podemos usar la superglobal `$_SESSION` para el almacenamiento de información:



### VIDEO DE INTERÉS

En el vídeo encontrarás una explicación de creación y uso de `$_SESSION`



<https://youtu.be/eW23BMX3eMM>



### ENLACE DE INTERÉS

En el siguiente enlace tienes el código utilizado:



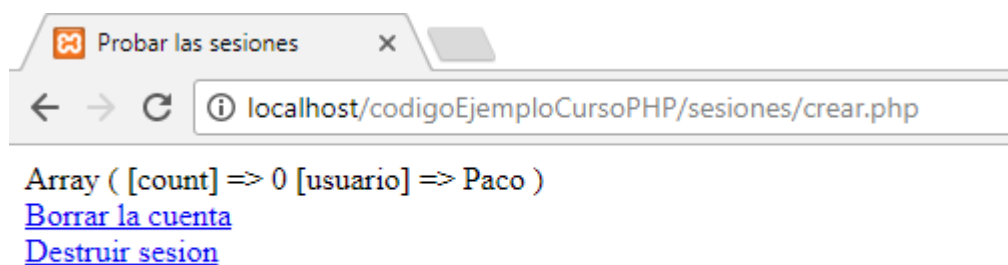
<https://github.com/pacogomezarnal/codigoEjemploCursoPHP/tree/master/sesiones>



Como podemos ver, en el código y en el vídeo:

### Crear sesión

```
<?php
    session_start();
    if (!isset($_SESSION['count'])) {
        $_SESSION['count']=0;
        $_SESSION['usuario']="Paco";
        print_r($_SESSION);
    } else {
        $_SESSION['count']++;
        print_r($_SESSION);
    }
?>
```



**Imagen 1b: Resultado de ejecución.** Fuente de la imagen: propia.

1. Para activar la sesión desde PHP y por lo tanto unir la sesión del usuario con la sesión almacenada en el servidor debemos utilizar la función `session_start()`; Si no utilizamos esta función, la sesión no se relacionará con el cliente.
2. Utilizando el array `$_SESSION`, podremos añadir, modificar y/o eliminar datos a dicha sesión.

## EJEMPLO PRÁCTICO

Una vez que hemos visto el inicio de creación de las sesiones:



1. Crearemos dentro de una estructura de directorios dentro de nuestro servidor un nuevo fichero denominado numVisitas.php
2. Iniciamos la sesión con `sesión_start`
3. Añadimos una variable numVisitas dentro de la superglobal `$_SESSION`
4. Añadimos 1 siempre y cuando la sesión esté iniciada.

### 1.3. Destruyendo sesiones

Hemos antes dicho que una de las tres formas que teníamos para destruir una sesión era siempre desde el servidor, bien porque el usuario nos indique que quiere eliminar la sesión y toda la información relacionada, o bien porque la aplicación así lo requiera.

#### VIDEO DE INTERÉS



En el vídeo encontrarás una explicación de destrucción de la sesión.



<https://youtu.be/FHGRk31OS8A>

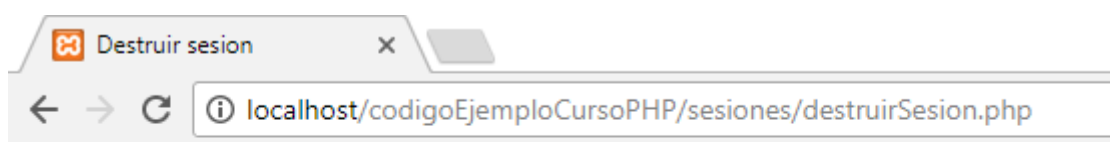
### ENLACE DE INTERÉS



En el siguiente enlace tienes el código utilizado:



<https://github.com/pacogomezarnal/codigoEjemploCursoPHP/tree/master/sesiones>



Array ( )

**Imagen 2b: Resultado de ejecución.** Fuente de la imagen: propia.

El primer y gran uso que le damos al uso de las sesiones es integrarlo con el control de acceso a las aplicaciones, los roles y la seguridad. Aunque encontrareis infinidad de código e información al respecto sobre el control de acceso y ejemplos con php, vamos a dar las pautas más importantes para la incorporación de las sesiones al sistema de seguridad.

## 1.4. PASO 1: Objeto Seguridad

El primer paso será seguir el modelo orientado a objetos que estamos utilizando a lo largo de todo el contenido y generar una nueva clase que controle y gestione la parte de sesión. Esta clase se encontraría dentro de lo que se denomina Controlador en el Patrón MVC.

Si entrar a describir por completo la clase las partes importantes que tendría serían:

### El Constructor

```
function __construct()  
{  
    //Arrancamos la sesion  
    session_start();  
    if(isset($_SESSION["usuario"])) $this->usuario=$_SESSION["usuario"];  
}
```

Como vemos, al concentrar en una sola clase el manejo de las sesiones, colocaremos en el constructor de esta clase la activación de la clase, de esta forma se realizará siempre y sólo se realizará una vez.

### Usando la clase

```
$seguridad=new Seguridad();
```

Al tener la clase definida, la utilización será muy sencilla, creando un nuevo objeto allí donde se vaya a necesitar.

## 1.5. PASO 2: El formulario de registro

El formulario de registro puede ser tan sencillo como el que se muestra en la imagen o tan complejo como la aplicación necesite información del usuario:

Formulario de registro

Email

Contraseña

Confirmar contraseña

☒ Deseo recibir noticias, actualizaciones de software, y la última información relativa a productos y servicios.

Registrarse

**Imagen 4: Formulario de registro.** Fuente de la imagen:

<http://blog.openalfa.com/wp-content/uploads/sites/2/2014/11/formulario-de-registro.jpg>

Una de las particularidades que vemos en el anterior formulario y que sucede en un 100% de formularios de registro es la aparición de un doble campo de comprobación de la contraseña de registro del usuario. Esto se consigue a nivel de HTML de la siguiente forma:

Doble campo de contraseña en el formulario

```
<label for="pass0">Contraseña</label>
<input type="password" id="pass0" name="pass0"
placeholder="Contraseña..">

<label for="pass1">Repite Contraseña</label>
<input type="password" id="pass1" name="pass1"
placeholder="Contraseña..">
```

Como observamos, los campos son de tipo password, pero lo importante en este caso es que el *name* de ambos campos sea diferente para poder recogerlos mediante `$_POST` y poderlos comparar.

### COMPRUEBA LO QUE SABES



Una vez visto el formulario de registro ¿serías capaz de añadir el campo usuario además del de email para ser almacenado?

Coméntalo en el foro.

## 1.6. PASO 3: La codificación de la contraseña

Como observamos en el anterior ejemplo la contraseña es texto plano que recibimos a través de los campos *pass0* y *pass1*. ¿Cómo se codifican dichas contraseñas en nuestra base de datos para que sea seguro?


Utilizaremos funciones ya programadas en php tipo hash como es sha o md5.

## ARTÍCULO DE INTERÉS



Seguramente todos conocemos y usamos stackoverflow para la resolución de nuestros problemas incluso como propuesta de código realizado.

Aquí tenemos otro uso de stackoverflow como fuente de información (siempre que sepas filtrar y seguir informándonos al respecto)

 <https://stackoverflow.com/questions/16713810/how-secure-is-md5-and-sha1>

### Ejemplo sha

```
<?php
$str = 'apple';

if (sha1($str) === 'd0be2dc421be4fcd0172e5afceea3970e2f3d940') {
    echo "Would you like a green or red apple?";
}
?>
```

### Ejemplo md5

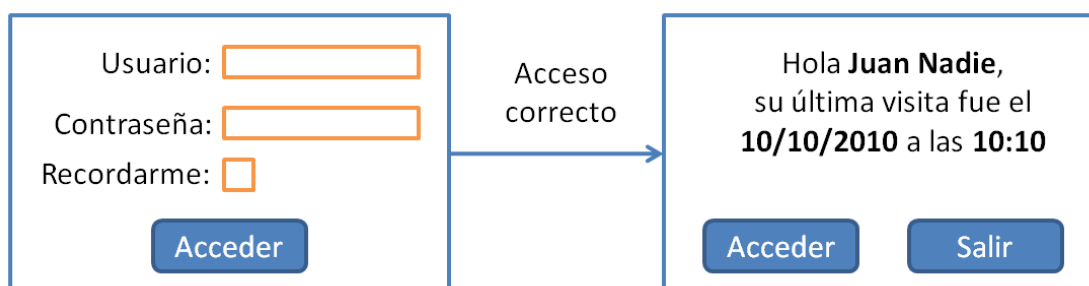
```
<?php
$str = 'apple';

if (md5($str) === '1f3870be274f6c49b3e31a0c6728957f') {
    echo "Would you like a green or red apple?";
}
?>
```

Como vemos, el resultado es muy parecido, produciendo en ambos casos hash que ahora sí almacenaremos en la base de datos. A su vez en la base de datos tendremos preparado un campo para almacenar el password que será de tipo VARCHAR si estamos trabajando con MySQL.

## 1.7. PASO 4: El formulario de acceso

El Formulario de acceso nos permite recoger la información de un usuario ya registrado en la base de datos y comprobar que existe en la misma con dicho usuario y dicha password. El único proceso a tener en cuenta en este paso es que en el momento de comprobar la contraseña, la misma se encuentra codificada en la base de datos, normalmente con un sistema de encriptación tipo MD5 o SHA, tal y como hemos visto en el PASO3.



**Imagen 5: Formulario de login.** Fuente de la imagen:

<http://idesweb.com/img/proyecto/prac09/acceso-recordatorio.png>

Así pues, en este punto:

1. Recogeremos los datos del formulario.
2. Recogeremos los datos del usuario en la BBDD.
3. Comprobaremos la contraseña almacenada con la contraseña que hemos recogido del formulario (recordemos que esa contraseña está como texto plano y por lo tanto deberemos aplicarle la misma función de codificación que en el caso de registro, sha1 o md5).



## COMPRUEBA LO QUE SABES



Una vez visto el formulario de acceso ¿serías capaz de utilizar el campo usuario o email dependiendo de lo que rellene el usuario?

Coméntalo en el foro.

## 1.8. PASO 5: Activación de la sesión

Una vez que el usuario se encuentra logado ya que lo hemos comprobado en el paso anterior, deberemos activar la sesión y añadir el usuario a dicha sesión:

### Ejemplo md5

```
function __construct()
{
    //Arrancamos la sesion
    session_start();
    if(isset($_SESSION["usuario"])) $this->usuario=$_SESSION["usuario"];
}

public function getUsuario(){
    return $this->usuario;
}

public function addUsuario($usuario,$pass,$remember=false){
    //GGenerando la variable de sesion
    $_SESSION["usuario"]=$usuario;
    $this->usuario=$usuario;
    //Almacenaremos el user/pass cookies
    if($remember)
    {
        setcookie("usuario",$usuario,time()+(60*60));
        setcookie("pass",$pass,time()+(60*60));
    }
}
```

Vemos en el anterior código varias funciones que intervienen en este paso:

- La primera ya la habíamos visto anteriormente ya que el constructor activa la sesión y comprueba si el \$\_SESSION["usuario"] se encuentra activo.
- addUsuario permite añadir el usuario a la sesión una vez que se ha logado.
- getUsuario permite recoger el usuario.

## 1.9. PASO 6: Proteger una página

Una vez que tenemos implementado todo el mecanismo anterior es muy sencillo proteger cualquier página con el siguiente código:

### Proteger página

```
<?php
include "../lib/Seguridad.php";
$seguridad=new Seguridad();
if($seguridad->getUsuario()==null){
    header('Location: login.php');
    exit;
}
?>
```

Como vemos, antes de cargar absolutamente nada de la página a proteger, antes de la etiqueta <HTML>, introducimos el anterior código y en el que:

- Primero incluimos la clase Seguridad y creamos el objeto (por lo que se activa la sesión).
- Comprobamos que exista el usuario.
- Si no existe redirigimos a una zona de la aplicación no protegida.

### COMPRUEBA LO QUE SABES



Una vez visto el acceso y las sesiones ¿serías capaz de mostrar por pantalla el usuario que se ha registrado a tu aplicación?

Coméntalo en el foro.

## 1.10. PASO 7: Logout

Realizar el logout es muy sencillo ya que únicamente debemos crear una función dentro de la clase seguridad que destruya la información dentro de la sesión:

### Logout

```
public function logout () {  
    $_SESSION=[];  
    session_destroy();  
}
```

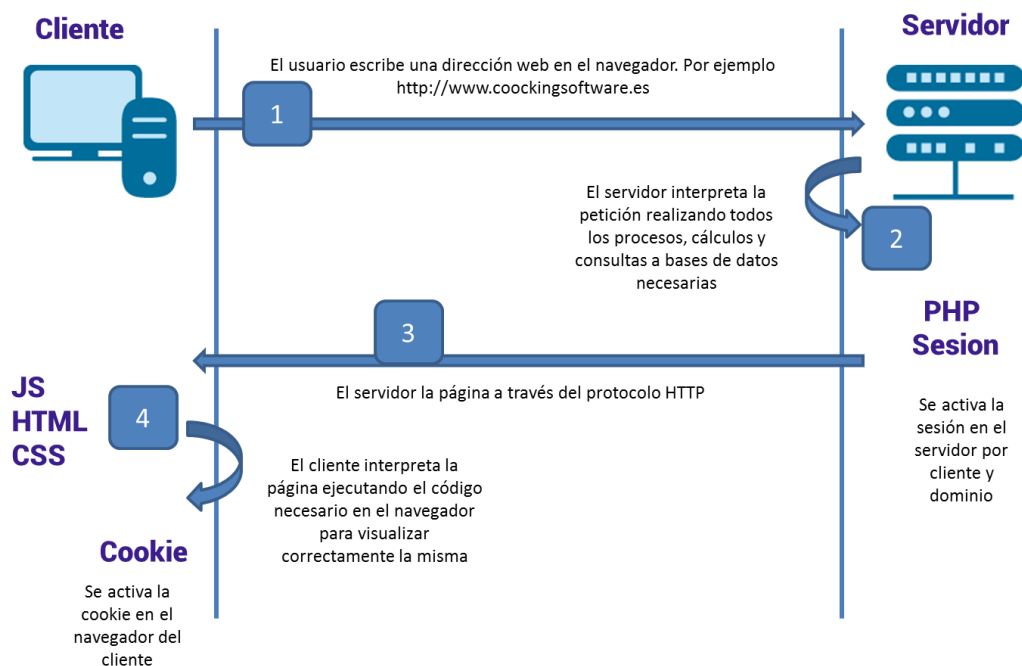
## 2. COOKIES

Mediante las sesiones, podemos generar información particular/individual de un usuario que mantenemos lo que dure la sesión abierta. Es por lo tanto una información temporal que perderemos o bien cuando el usuario cierre la sesión, o bien cuando queramos borrar dicha sesión borrando todos los datos dentro de la sesión generados.

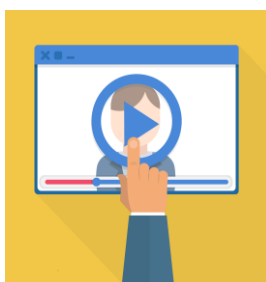
Tenemos un segundo mecanismo de interacción con el navegador, utilizar las cookies. Según la definición de php net, las cookies son un mecanismo por el que se almacenan datos en el navegador remoto para monitorizar o identificar a los usuarios que vuelvan al sitio web. En este caso, SI almacenamos información en el equipo del usuario, y por lo tanto aunque el usuario cierre el navegador dicha información permanecerá durante cierto tiempo esto tiene sus partes positivas y sus partes negativas:

- Tenemos un mecanismo sencillo para poder almacenar información y además que sea dependiente del usuario que realiza la petición.
- La información podrá ser alcanzada por cualquier otro software, y por lo tanto NO debemos almacenar información personal, sensible y que pueda ser utilizada posteriormente para el uso malicioso contra el usuario.

Recordemos el diagrama que teníamos al principio de la Unidad:



**Imagen 3: Cookies vs sesiones.** Fuente de la imagen: propia.



### VIDEO DE INTERÉS

En el vídeo encontrarás una explicación sobre las cookies.



<https://youtu.be/IsHEBpZpSJA>

## 2.1. Creando y recuperando cookies

Crear una nueva cookie y recuperar esta va a ser una tarea tremendamente sencilla. Para ello seguiremos las indicaciones y el ejemplo que la misma documentación oficial de php.net no ofrece:

### Manejo de cookies

```
<?php
Setting new cookie
=====

<?php
setcookie("name", "value", time()+$int);
/*name is your cookie's name
value is cookie's value
$int is time of cookie expires*/
?>

Getting Cookie
=====

<?php
echo $_COOKIE["your cookie name"];
?>

Updating Cookie
=====

<?php
setcookie("color", "red");
echo $_COOKIE["color"];
/*color is red*/
/* your codes and functions*/
setcookie("color", "blue");
echo $_COOKIE["color"];
/*new color is blue*/
?>

Deleting Cookie
=====

<?php
unset($_COOKIE["yourcookie"]);
/*Or*/
setcookie("yourcookie", "yourvalue", time()-1);
/*it expired so it's deleted*/
?>

?>
```



### VIDEO DE INTERÉS

En el vídeo encontrarás una explicación sobre el uso de las cookies.



<https://youtu.be/lsAJFPyG1AQ>



### ENLACE DE INTERÉS

En el siguiente enlace tienes el código utilizado:



<https://github.com/pacogomezarnal/codigoEjemploCursoPHP/tree/master/cookies>



**Imagen 3b: Resultado de ejecución.** Fuente de la imagen: propia.

Como podemos observar en los ejemplos y en la documentación, una cookie permanecerá activa en nuestro navegador el tiempo que hayamos programado.

## EJEMPLO PRÁCTICO



Una vez que hemos visto el inicio de creación de las cookies:

1. Crearemos dentro de una estructura de directorios dentro de nuestro servidor un nuevo fichero denominado numVisitas.php
2. Iniciamos la cookie numVisitas con el valor 0.
3. Añadimos 1 siempre y cuando la cookie exista.

## 2.2. Aplicación de las cookies al login

Un uso de las cookies muy típico es dentro del formulario de login de acceso a nuestra aplicación. Adelanto que la implementación realizada es MUY INSEGURA, ya que almacena el usuario y la contraseña, pero que nos sirve para comenzar a utilizar las cookies dentro de nuestras aplicaciones:



### VIDEO DE INTERÉS

En el vídeo encontrarás una explicación sobre el uso de las cookies.



<https://youtu.be/TwfMR47i-98>



### ENLACE DE INTERÉS



En el siguiente enlace tienes el código utilizado:



<https://github.com/pacogomezarnal/codigoEjemploCursoPHP/tree/master/seguridad>

### COMPRUEBA LO QUE SABES



Una vez visto el uso de sesiones y cookies ¿serías capaz de crear un sistema de control de visitas por página y que un mismo usuario no contabilice dos veces la misma página?

Coméntalo en el foro.

## Resumen final:

En esta unidad hemos aprendido y utilizado dos de los conceptos más importantes en el desarrollo de aplicaciones web: sesiones y cookies.

Mediante la combinación de ambos conceptos podemos implementar la seguridad de usuario y la particularización de los desarrollos. Con las sesiones y las cookies podemos desarrollar muchas otras acciones y utilidades a partir de los conocimientos aprendidos.

Las sesiones se inician, gestionan y finalizan desde el servidor, por lo que nos proporciona un mecanismo para poder realizar la autenticación y gestión de la seguridad. Las cookies se gestionan desde el cliente, por lo que son más inseguras, pero por otro lado proporcionan la permanencia de datos tan importante para almacenar información con la dupla usuario-navegador.