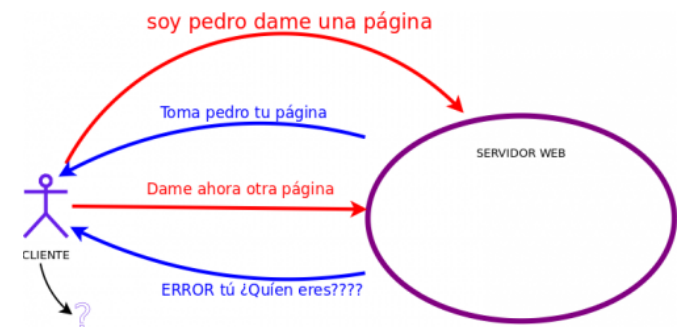


# UT4

## DESARROLLO DE APLICACIONES UTILIZANDO CÓDIGO EMBEBIDO COOKIES Y SESIONES

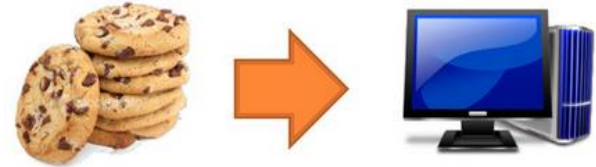
# Introducción

- HTTP es un **protocolo sin estado**, lo que significa que cada conexión es completamente **independiente** de las demás.
- Esto implica que con HTTP *no se puede guardar información que relacione una conexión con otra*. Por tanto dificulta **seguir la pista** a las acciones de los usuarios.
- Para mantener los valores de las variables PHP a lo largo de toda la navegación (*ejecución de varios scripts*) usamos:
  - **Cookies**: los valores de las variables generadas en el servidor se almacenan en el **cliente**.
  - **Sesiones**: los valores de las variables generadas en el servidor se almacenan en el **servidor**.



# 1. COOKIES: Introducción

- Una **cookie** es un *pequeño archivo de información* sobre el usuario, que el servidor coloca en el equipo cliente con diferentes fines:



- **Autenticar** al *usuario* de forma directa, es decir, solo la primera vez es necesario escribir el *login* y la *contraseña*.
- **Diferenciar** a los *usuarios* y ofrecer a cada uno el contenido que corresponda.
- Establecer las **preferencias** y **opciones** del *usuario*: por ejemplo, métodos de búsquedas de Google, idioma del usuario, etc.

# 1. COOKIES: Introducción

- ❑ Fueron desarrolladas por primera vez en **1994** por **Netscape**.
- ❑ Actualmente quedan referenciadas en el **documento RFC 6265** (*HTTP State Management Mechanism*) de *IETF* de abril 2011.
- ❑ La información se almacena a petición del servidor web:
  - ❑ Directamente desde la propia página con *JavaScript*  
**document.cookie**
  - ❑ Desde el servidor web mediante las cabeceras HTTP  
**PHP → setcookie() y \$\_COOKIE**

# 1. COOKIES: Introducción

- ❑ Por defecto, se mantienen “vivas” mientras el cliente web está abierto.
  - ❑ Sin embargo, se pueden configurar para que “vivan” más allá de la sesión del navegador.
- ❑ Solamente son visibles por el servidor que las creó.
- ❑ El usuario puede decidir deshabilitarlas o eliminarlas o modificarlas en el cliente web, mediante la opción de *Borrar datos de navegación*.

Borrar datos de navegación

Eliminar elementos almacenados desde: el origen de los tiempos

- ☒ Historial de navegación
- ☒ Historial de descargas
- ☒ Cookies y otros datos de sitios y de complementos
- ☒ Archivos e imágenes almacenados en caché
- ☒ Contraseñas
- ☐ Datos de Autocompletar formulario
- ☒ Datos de aplicaciones alojadas
- ☒ Licencias de contenido

[Más información](#)

Borrar datos de navegación Cancelar

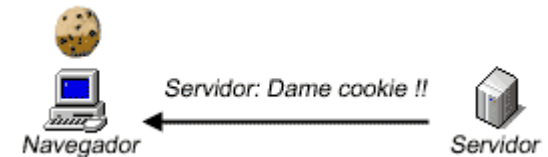
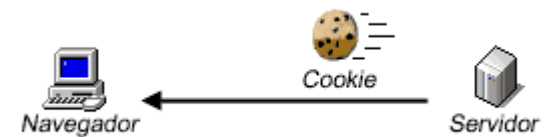
La información de configuración de contenido y motores de búsqueda guardada no se borrará y podría reflejar tus hábitos de navegación.

# 1. COOKIES: Introducción

- ❑ Las cookies **no son**:
  - ❑ Software
  - ❑ Fragmentos de código.
- ❑ Las cookies **son** solamente **datos**
  - ❑ Por tanto no pueden transmitir ni ejecutar virus ni tampoco instalar malware como troyanos ni programas de espionaje.
- ❑ Sin embargo, **sí** que pueden utilizarse para realizar un **seguimiento** del usuario en la web mediante “analítica web” (*tracking cookies*).

# 1. COOKIES: Funcionamiento

- ❑ El usuario solicita, mediante el navegador, una página al servidor.
- ❑ En la respuesta, la cookie es **enviada al navegador del cliente** desde el servidor y, si éste la acepta, permanece en él.
- ❑ Las **páginas piden la cookie** al navegador.
- ❑ El **navegador envía la cookie**, permitiendo la identificación del usuario por parte del servidor.



# 1. COOKIES: Ejemplos de aplicación

- ❑ Memorizar si un usuario visitó un producto o categorías de producto y mostrarle publicidad a medida.
- ❑ Un usuario accedió a un formulario de ingreso pero no lo usó o no lo completó. Se le puede animar a completar la inscripción.
- ❑ Igualmente, si un usuario accede a una aplicación de compra de producto, pero no completó la compra.
- ❑ Se puede guardar la fecha de la última visita de un usuario al sitio web. De esta manera, calculando la diferencia entre la fecha actual y la fecha de la última visita se pueden añadir enlaces que apunten hacia contenidos publicados posteriores a esa última visita.



# 1. COOKIES: Seguridad

- ❑ Las cookies se asocian con una combinación **máquina+usuario\_del\_sistema\_operativo+navegador**.
- ❑ Por tanto, son accesibles para todas la personas que compartan esta combinación, de modo que es muy recomendable **no guardar información sensible como:**
  - ❑ Datos personales, contraseñas, etc., según establece la ***LOPD*** (*Ley Orgánica de Protección de Datos*).
  - ❑ Además, la UE obliga actualmente a avisar a los clientes del uso de cookies en un sitio web.
- ❑ Cada cookie está formada por una pareja  
**("nombre", "contenido")** de tipo texto.

# 1. COOKIES: Seguridad

- ❑ Las cookies pueden contener información sobre los **hábitos de navegación** de los usuarios en un sitio web, habitualmente para ser utilizados con **finés publicitarios**.
- ❑ Además, un sitio puede **redireccionar** a otros servidores (**de forma transparente al cliente**) para que éstos creen sus propias cookies, lo que se conocen como **cookies de terceros**. Por eso, ciertos sitios nos ofertan productos que hemos visitado anteriormente en otro sitio web
  - ❑ Aunque esta opción se puede desactivar al crear la cookie.

# 1. COOKIES: Información relevante

- ❑ Nombre (sin caracteres acentuados) / Contenido (textual)
- ❑ Fecha de expiración o caducidad
- ❑ Dominio
- ❑ Ruta
- ❑ Transmisión por protocolo seguro
- ❑ Bloqueo a lenguajes de script en el cliente.
  - ❑ `ini_set('session.cookie_httponly', 1);`

▼ Cabeceras de la respuesta (0,353 KB)	
Connection:	"Keep-Alive"
Content-Length:	"326"
Content-Type:	"text/html; charset=UTF-8"
Date:	"Wed, 21 Oct 2015 21:34:21 GMT"
Keep-Alive:	"timeout=5, max=100"
Server:	"Apache/2.4.12 (Win32) OpenSSL/1.0.1i PHP/5.6.8"
Set-Cookie:	"visita=2015-10-21T23%3A34%3A21%2B02%3A00; expires=Wed, 21-Oct-2015 22:34:21 GMT; Max-Age=3600"
X-Powered-By:	"PHP/5.6.8"

# 1. COOKIES: Localización y manejo

- ❑ En el navegador *Google Chrome* podemos verlas y administrar su uso desde o con **EditThisCookie**:



<chrome://settings/configuración de contenido/cookies>

- ❑ Se puede hacer un listado de las cookies por servidor, consultar su contenido, borrarlas, y de manera global deshabilitarlas o habilitarlas.

# 1. COOKIES: Creación de cookies en PHP

- ❑ Las cookies se envían en las cabeceras HTTP, y deberemos **generarlas con `setcookie`** antes de enviar ningún otro dato al cliente.
- ❑ El manejo de las cookies en PHP es extremadamente sencillo:
  - ❑ En el 1<sup>er</sup> paso se **envía** la cookie:
    - llamar a la función **`setcookie`** para crear la cookie en el cliente
  - ❑ En las posteriores peticiones que recibamos de ese cliente vendrá incrustada la cookie => se **consulta** la información almacenada en ella
    - acceder mediante la variable superglobal **`$_COOKIE`**.

```
<?php  
    var_dump($_COOKIE);  
?>
```

# 1. COOKIES: Creación de cookies en PHP

- ❑ Sintaxis: **setcookie ("nombre", ["contenido"], [...], )**
- ❑ Argumentos de la función **setcookie**
  - ❑ "nombre" = nombre de la cookie
  - ❑ ["contenido"] = contenido de la cookie (puede ser vacío)
  - ❑ ["caduca"] = fecha y hora de caducidad expresado en tiempo Unix; por defecto, la cookie se borra al cerrar la ventana del navegador
  - ❑ ["ruta"] = establece el directorio del servidor al que se asocia la cookie, que será enviada cuando el cliente acceda a dicho directorio (o subdirectorios)
  - ❑ ["dominio"] = establece el dominio del servidor
  - ❑ ["enviar para"] = es un booleano que indica si la cookie se enviará en conexiones seguras https o en cualquier tipo de conexión
  - ❑ ["accesible"] = establece si la cookie será accesible solo por el servidor o mediante el navegador usando scripts.

# 1. COOKIES: Creación de cookies en PHP

## □ Argumentos de la función **setcookie**

`bool setcookie (string nombre, [string valor], [int expirer], [string ruta],  
string [dominio], bool [segura], bool [httponly]);`

## Ejemplos de cookies

```
<?php  
setcookie("saludo");  
?>
```

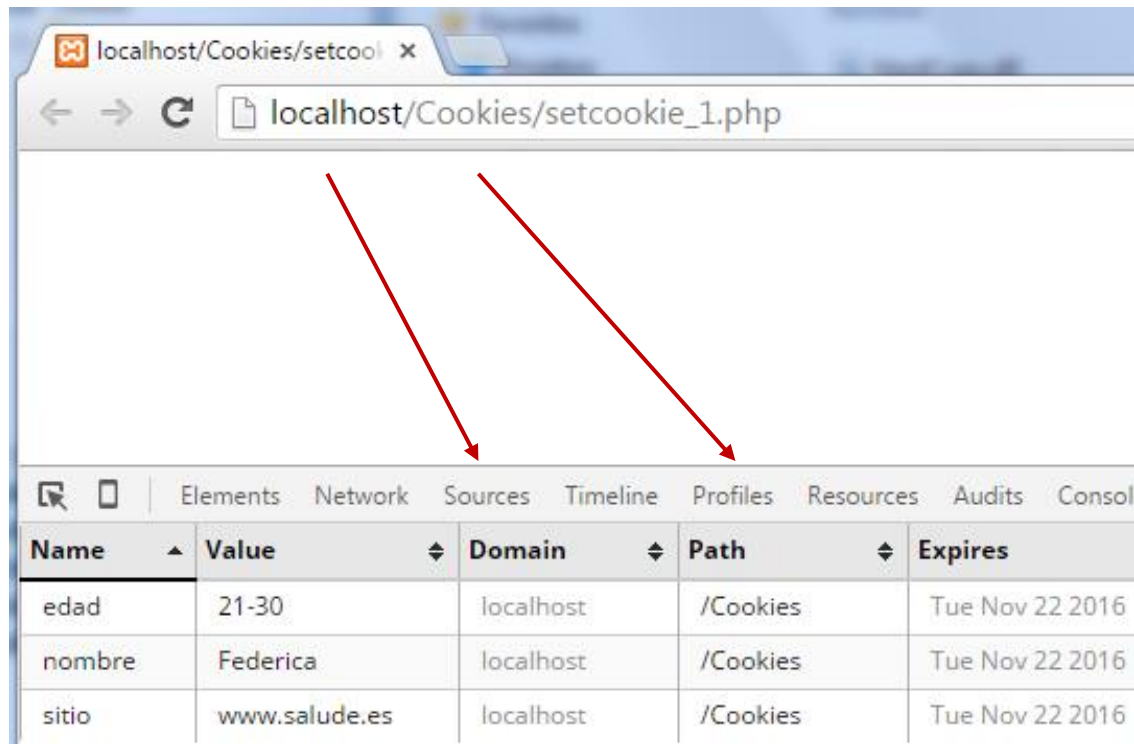
```
<?php  
setcookie("saludo", "hola");  
?>
```

```
<?php  
// Esta cookie dura una hora → 1 hora = 3600 segundos  
setcookie("visitas", "1", time()+3600);  
?>
```

```
<?php  
// Esta cookie dura un año  
setcookie("visitas", "1", time()+60*60*24*365);  
?>
```

# 1. COOKIES: Ejemplo en PHP

```
<?php
setcookie ('nombre', 'Federica');
setcookie ('edad', '21-30');
setcookie ('sitio', 'www.salude.es');
?>
```



The screenshot shows a web browser window with the address bar displaying 'localhost/Cookies/setcookie\_1.php'. Below the browser window, the developer tools are open to the 'Sources' tab, showing a table of cookies. The table has columns for Name, Value, Domain, Path, and Expires. The cookies listed are 'edad' (21-30), 'nombre' (Federica), and 'sitio' (www.salude.es). Two red arrows point from the PHP code above to the 'edad' and 'sitio' rows in the table.

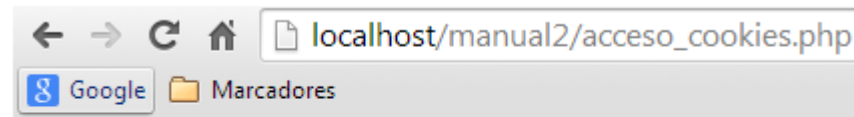
Name	Value	Domain	Path	Expires
edad	21-30	localhost	/Cookies	Tue Nov 22 2016
nombre	Federica	localhost	/Cookies	Tue Nov 22 2016
sitio	www.salude.es	localhost	/Cookies	Tue Nov 22 2016



# 1. COOKIES: Acceso a las cookies en PHP

- ❑ En PHP se puede acceder a las cookies mediante las variables superglobales **\$\_COOKIE** y **\$\_REQUEST**.
- ❑ La función **isset(argumento1, [argumento2], ...)** indica si la(s) cookie(s) está(n) activada(s). Devuelve un booleano:
  - ❑ **true** si el argumento(s) existe(n) y no es NULL
  - ❑ **false** en caso contrario.

```
<?php
if (isset($_COOKIE['nombre'])) {
    echo "<p>Hola de nuevo $_COOKIE[nombre]</p>\n";
    if (isset($_COOKIE['sitio'])) {
        echo "<p>Tu sitio web preferido es $_COOKIE[sitio] </p>\n";
    }
} else {
    echo "<p>Bienvenid@ a nuestro portal</p>\n";
}
?>
```



# 1. COOKIES: Modificación de cookies en PHP

---

- ❑ Para *modificar* una cookie en PHP, no hay forma de hacerlo.
- ❑ En realidad, lo que se hace es establecerla de nuevo con **setcookie**.

# 1. COOKIES: Borrado de cookies en PHP

- ❑ Para *eliminar* una cookie en PHP tan sólo debemos volver a **crearla** indicando un *instante anterior al actual*.

```
01. <?php
02.     if( isset($_COOKIE['nombre']) )
03.     {
04.         echo "Eliminamos la Cookie";
05.         setcookie("nombre", "", time() - 1 );
06.     }
07.     else
08.     {
09.         echo "No existe la Cookie";
10.     }
11.
12.     echo "<p><a href='05_cookies_eliminar_2.php'>Haz clic para recargar la página y comprobar si exist
    e la Cookie</a></p>";
13. ?>
```

Si usáramos la función de PHP **unset()**, únicamente la borraríamos del array asociativo **\$\_COOKIES** pero continuaría en el navegador web del usuario, y al recargarse la página o ser cargada otra vez se volvería a recuperar su valor.

## 2. SESIONES: Introducción

- Una **sesión** es la *secuencia de páginas que un usuario visita en un sitio web*. Es decir, desde que **entra** en el sitio, hasta que lo **abandona**.



- El uso de sesiones en PHP es un método ampliamente extendido.
- Tiene diversas utilidades, pero sin duda la más común es el control de acceso a usuarios, aunque debemos pensar que **no son sinónimos**.

## 2. SESIONES: Introducción

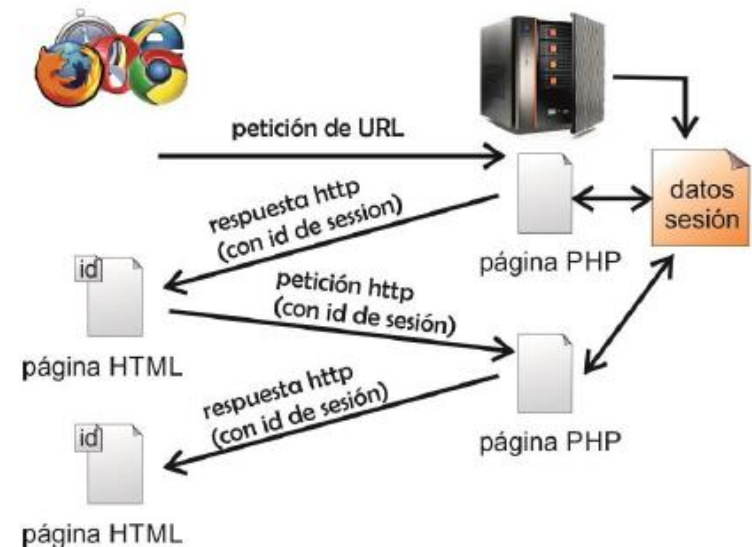
- ❑ En una aplicación web se navega por varias páginas. Y dicha aplicación utiliza el protocolo HTTP. Así pues:
- ❑ ¿ Cómo guardar los datos que introduce el usuario en cada una de las páginas ?
- ❑ ¿ Cómo pasar la información para obtener la salida final ?
- ❑ **Objetivo:** Garantizar la conservación de los datos de un cliente que está realizando una operación en la Web.

## 2. SESIONES: Introducción

- ❑ Al crear una variable en un script PHP, ésta se encontrará disponible durante la ejecución de la página contenida en un archivo *.php*, eliminándose automáticamente una vez finalizado el **script**.
- ❑ Sin embargo, en ocasiones necesitaremos que determinada información esté disponible en diferentes páginas en PHP y en posteriores accesos a las mismas. Para ello podemos usar **variables de sesión**.
- ❑ Las **sesiones** nos sirven para almacenar información que se memorizará durante toda la **visita de un usuario** a una página web.

## 2. SESIONES: Introducción

- ❑ Las sesiones son un sistema que permite a los servidores guardar una relación entre conexiones, pero a diferencia de las cookies, la **información se almacena en el servidor**, no en el navegador web.
- ❑ Además, esta información se guarda durante un **tiempo determinado**, que puede ser el cierre de sesión por parte del usuario o el tiempo de expiración de una sesión establecida previamente.

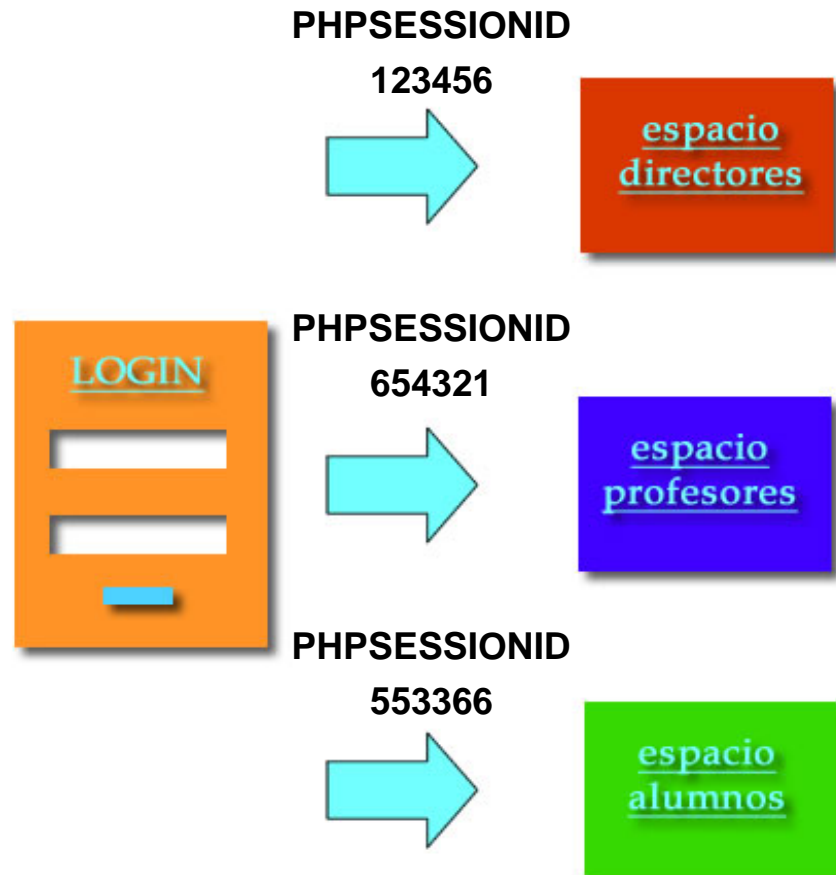


## 2. SESIONES: Introducción

- Para cada usuario, se genera internamente un **identificador de sesión único** que sirve para saber las variables de sesión que pertenecen a cada usuario.

**PHPSESSIONID**

**JSESSIONID**  
**ASPSESSIONID**





## 2. SESIONES: Ciclo de vida

- ❑ Para que las sesiones funcionen, no pueden aparecer páginas estáticas (documentos HTML) en pasos intermedios.
- ❑ El manejo de las sesiones se realiza de la siguiente forma:
  - ❑ Todas las páginas de tipo servidor deben realizar una llamada a **session\_start()** para cargar las variables de la sesión
  - ❑ *Esta llamada debe estar colocada antes de cualquier código HTML*
  - ❑ Conviene llamar a **session\_destroy()** para cerrar la sesión.

## 2. SESIONES: Creación o inicio

- ❑ Las sesiones *no se activan por sí mismas*, sino a través de la ejecución de una función PHP (por tanto, en la parte servidor): **session\_start**
- ❑ Dicha ejecución puede programarse para que suceda por el hecho de llegar a una página, o mediante **alguna acción** específica del usuario:
  - ❑ Por ejemplo, en un catálogo de comercio electrónico, por el mero hecho **de entrar** a la página de inicio del sitio.
  - ❑ En otros casos, no se inicia la sesión hasta que el usuario **haga algo**. Por ejemplo, escribir usuario y contraseña en el envío de un formulario de acceso. Es el caso de los sitios de *banca on-line*, *campus virtuales*, *webmail*, etc.

## 2. SESIONES: Creación o inicio

- ❑ La función **session\_start** comprueba si hay una sesión abierta:
  - ❑ Si no la hay, **creará** una
  - ❑ Si la hay, **reanudará** la sesión con el mismo ID.
- ❑ Cuando un script PHP crea una sesión, el servidor asocia el navegador del usuario con el archivo de sesión ubicado en el servidor.
- ❑ El identificador se guarda en el usuario en forma de cookie; *si no se permitiera el uso de cookies se añadiría el ID en la dirección de la página.*

## 2. SESIONES: Creación o inicio

- Igual que con las cookies, el ID de sesión se envía con las cabeceras.
- **Por tanto, la función `session_start()` debe ir antes del contenido de la página (*incluso antes de llamadas a funciones con `include()` y `require()`*).**

```
<?php
session_start();
echo 'Sesión iniciada con ID = '.session_id();
?>
```

Debe ir antes de cualquier generación de salida.

## 2. SESIONES: Creación o inicio

```
<?php
session_start();
echo 'Sesión iniciada con ID = '.session_id();
?>
```

The screenshot shows a web browser window with the address bar displaying `localhost/Sesiones/inisession.php`. The page content shows the output of the PHP script: "Sesión iniciada con ID = ft4v10bbi6ud14ptn664gab555". The ID is highlighted with a red box. A cookie manager window is open on the right, showing the details of the generated session cookie. The cookie name is `PHPSESSID`, its value is `ft4v10bbi6ud14ptn664gab555`, and its domain is `localhost`. The window also shows the path `/`, expiration date `22/11/2016 04:23 PM`, and various flags like `hostOnly`, `sesión`, `Seguro`, and `httpOnly`. A green checkmark is visible at the bottom of the cookie manager window, indicating the cookie is successfully set.

Sesión iniciada con ID = ft4v10bbi6ud14ptn664gab555

Cookie generada

valor  
ft4v10bbi6ud14ptn664gab555

Dominio  
localhost

Ruta  
/

caducidad  
22/11/2016 04:23 PM

hostOnly ☒ sesión ☒ Seguro ☐ httpOnly ☐

Ayuda

## 2. SESIONES: Registro

- ❑ Para permitir que una variable pertenezca a una sesión y pueda transmitirse entre diferentes páginas, es preciso registrarla mediante la variable superglobal **\$\_SESSION**:

**\$\_SESSION['variable'] = valor;**

- ❑ Para dejar de registrar una variable:

**unset(\$\_SESSION['variable']);**

- ❑ Para dejar de registrar todas las variables de sesión:

**session\_unset();**

## 2. SESIONES: Comprobación

- ❑ Se puede comprobar si una sesión está establecida mediante la función **isset['variable']**.
- ❑ Como la variable `$_SESSION` es superglobal, solamente se puede iniciar con `session_start()`.
- ❑ ¡¡ Por tanto, antes de comprobar si la sesión está establecida hay que invocar a `session_start()` !!;

```
<?php
    session_start();
    if ( isset($_SESSION['variable']) ) {
        // Aquí el código
    }
?>
```

```
<?php
    session_start();
    if ( !(isset($_SESSION['login']) ) {
        header("Location: login.php");
    }
?>
```



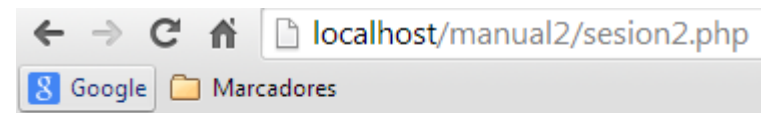
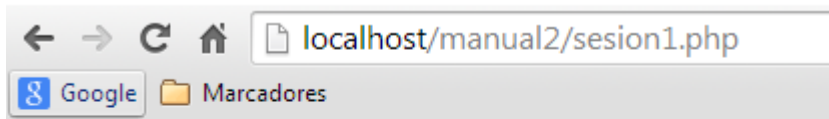
## 2. SESIONES: Registro. Ejemplo

```
<?php
session_start();
echo 'Sesión iniciada con ID = ' . session_id();
echo '<br>';
$nombre = 'Federica';
$_SESSION ['nombre'] = $nombre;
echo 'Variables de sesión: ';
var_dump ($_SESSION);
?>
<html> <br><a href="sesion2.php">sesion2.php</a> </html>
```

sesion1.php

```
<?php
session_start();
echo 'Sesión iniciada con ID = ' . session_id();
echo '<br> Hola de nuevo ' . $_SESSION ['nombre'];
?>
<html> <br><a href="sesion3.php">sesion3.php</a> </html>
```

sesion2.php





## 2. SESIONES: Destrucción o cierre

- ❑ Como se ha visto, la información relativa a las variables de una sesión en PHP se almacena en el *array asociativo* **\$\_SESSION**.
- ❑ Por tanto, una forma de "vaciar" la sesión, es decir, eliminar todas las variables de la misma, sería utilizar la instrucción genérica para inicializar un array: **\$\_SESSION = array();**
- ❑ Tanto el usuario como el servidor pueden cerrar la sesión.
  - ❑ Lo primero es dejar de registrar o liberar todas las variables con: **session\_unset()**
  - ❑ Después cerrar la sesión con: **session\_destroy()**.

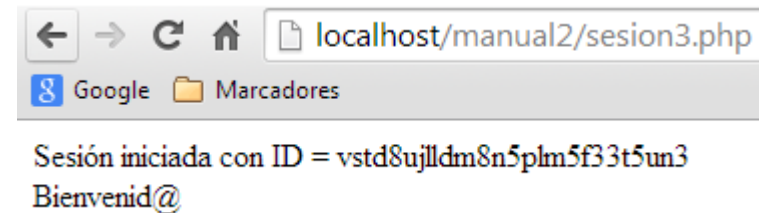
## 2. SESIONES: Destrucción o cierre. Ejemplo

```
<?php
session_start();           //recuperamos la sesión
session_unset();           //liberamos todas las variables
echo 'Sesión iniciada con ID = '.session_id();

if (isset ($_SESSION['nombre']))
    echo '<br> Hola de nuevo ' .$_SESSION ['nombre'];
else
    echo '<br> Bienvenid@ ';

session_destroy();         //cerramos la sesión
?>
```

sesion3.php



Como podemos observar la función **`session_unset()`** libera las variables pero **no cierra la sesión**, ya que se ve el mismo identificador de sesión. Para ello, será necesario utilizar **`session_destroy()`**.

## 2. SESIONES: Diferencias con las cookies

### Diferencias entre Session y Cookies

SESIONES	COOKIES
<ul style="list-style-type: none"><li>➤ No almacenan en el navegador del usuario.</li><li>➤ Usan Token de acceso y permite que la información que pasa la tenga hasta que su navegador este abierto.</li><li>➤ Cuando se cierra las sesiones se pierden.</li><li>➤ Se usan para transferir información e identificación de usuarios.</li><li>➤ Tienen un ciclo de vida.</li></ul>	<ul style="list-style-type: none"><li>➤ Almacenan en el navegador del usuario.</li><li>➤ Puede mantener información en el navegador del usuario hasta que se elimine.</li><li>➤ Cuando se cierra el navegador no se pierden.</li><li>➤ Se usan mas para el control de uso de password y usuarios.</li><li>➤ No tienen un ciclo de vida.</li></ul>