



UT3: DESARROLLO DE APLICACIONES WEB. COOKIES Y SESIONES

Néstor Sabater

1. PROTOCOLO HTTP

- Cuando nos conectamos a un sitio web remoto solemos hacerlo mediante el protocolo **http://**
 - **HTTP** es un protocolo de petición (request) y respuesta(response); básicamente, un cliente hace una petición de recurso y un servidor HTTP responde a esa petición.
 - Una conexión típica a Internet tiene la forma `http://www.nestorsb.com`. También podemos usar, en determinadas páginas, el protocolo `https://` (las llamadas páginas seguras).
-

1. PROTOCOLO HTTP

- Ambos son lo que se conoce como protocolos sin estado.
 - Esto significa que ni el servidor ni el cliente conservan memoria de los pares **nombre-valor** empleados en una página, al pasar a otra, a menos que pasemos dichos datos específicamente, mediante un formulario.
 - Para solventar esto aparecen las **cookies** y las **sesiones**.
-

1. PROTOCOLO HTTP

- Ambas son, en su concepto básico, formas de almacenar provisionalmente unos datos que otra página recopilará para seguir trabajando con ellos.
 - La diferencia fundamental es el lugar de almacenamiento de dichos datos.
 - Las cookies se usan para guardarlos en el cliente y las sesiones para guardarlos en el servidor.
-

1. PROTOCOLO HTTP

- Ambos sistemas tienen sus ventajas y sus limitaciones.
 - Ejemplos:
 - Si recurrimos a las sesiones, y nuestro sitio cuenta con muchas visitas simultáneas, el servidor se puede sobrecargar.
 - Si preferimos usar cookies y el cliente las tiene desactivadas, se presentará un fallo (solucionable, en última instancia).
 - Cuando necesitamos almacenar datos para pasarlos de una página a otra, el empleo de sesiones o de cookies es, en algunos casos, una cuestión de criterio personal.
 - En otras situaciones, debido al alcance que deban tener los datos, se presenta como más adecuado un sistema u otro. En este artículo aprenderemos a usar ambas técnicas y a decidir cuál elegir en cada caso.
-

2. COOKIES

- Una cookie (“galletita”, en inglés) es un archivo de texto que contiene el nombre de una variable y su valor.
 - Este archivo se almacena en el ordenador cliente durante la navegación por Internet y puede estar disponible para recuperar el valor de la variable en posteriores navegaciones.
 - Recuerda esto: una cookie solo puede contener texto plano, no programas ejecutables ni nada potencialmente peligroso.
 - Cuando el usuario se conecta a Internet y entra en nuestro sitio, el código en el servidor es el encargado de almacenar la cookie en el cliente.
-

2. COOKIES

¿Para qué queríamos almacenar una variable en el ordenador del cliente?

Puede haber tantas respuestas a esta pregunta como seas capaz de imaginar.

2. COOKIES

- Quizás uno de los usos más populares es el del idioma del propio usuario.
 - Tú creas un sitio web destinado a visitantes de diversas nacionalidades.
 - Para ello creas una página y luego haces una copia en la que cambias los textos a otro idioma.
 - Tienes, por tanto, dos copias de la página. (Español e inglés por ejemplo)
-

2. COOKIES

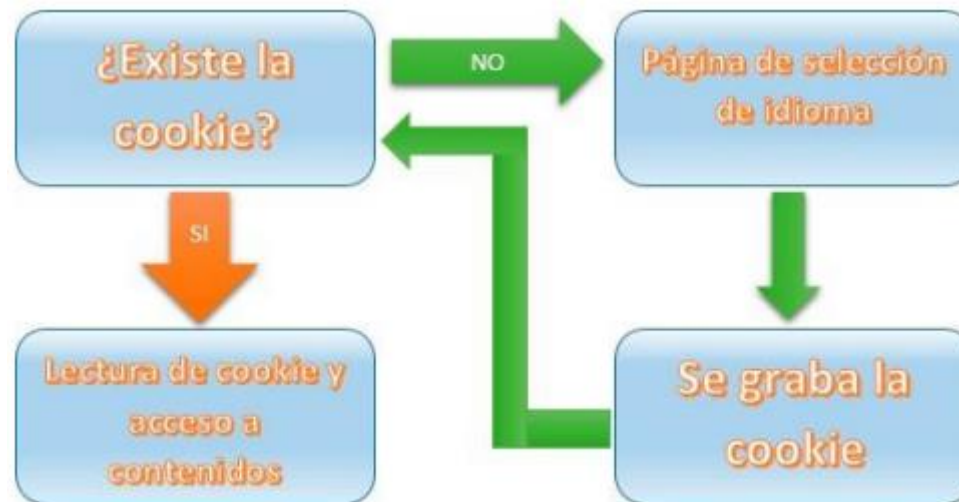
- Si un usuario se conecta por primera vez a tu sitio, tiene opción a elegir en qué idioma quiere ver los contenidos.
 - El nombre del idioma elegido queda almacenado en una variable en el propio ordenador del cliente (una cookie).
 - Ahora supongamos que el internauta se vuelve a conectar a tu sitio, digamos, al día siguiente. El servidor encuentra la cookie, lee el idioma que eligió el usuario y le remite, directamente, a la página en dicho idioma. Esto crea un efecto de personalización que contribuye a que la persona se sienta cómoda visitando el sitio.
-

2. COOKIES

- Veamos cómo usar este mecanismo.
 - En primer lugar, necesitamos un script que determine si el usuario se conecta a nuestro sitio por primera vez, o si ya se ha conectado antes, es decir, si en su ordenador no existe ninguna cookie nuestra que almacene su idioma o si ya está grabada.
 - En caso de que no exista la cookie, deberá cargarse una página donde se le pida al usuario su idioma y deberá crearse la cookie.
 - En el caso de que ésta ya exista, deberá leerse y redireccionar al usuario a la página en su propio idioma.
-

2. COOKIES

- El mecanismo responde al esquema de la siguiente imagen:



2. COOKIES

- Comprobando la cookie:

- Analiza el esquema.
- Verás que, en el acceso al sitio por la página principal, existe un script que verifica si el cliente que se ha conectado tiene grabada una cookie con una variable relativa al idioma del usuario.
- Si no se encuentra esta variable, se llega a una página donde se le pide que elija su idioma.
- Desde ahí, se salta a otra página donde se graba la variable y, por fin, a la página de contenidos elegida.
- Si existe la cookie, se lee la variable y, en función de su valor, se salta a la página elegida.



2. COOKIES

- Cuando un script debe leer una variable que está grabada en una cookie se recurre a una matriz propia de PHP, llamada **`$_COOKIE`**.
 - Se trata de una matriz asociativa cuyo índice es el nombre de la variable que esperamos poder leer desde la cookie del cliente.
 - En el ejemplo de las páginas con varios idiomas, el idioma del cliente podría, por ejemplo, almacenarse en una variable llamada `$idiomaUsuario`, con lo que para leer dicha variable desde una cookie usaríamos una llamada como la siguiente: `$_COOKIE["idiomaUsuario"]`.
-

2. COOKIES

- Para grabar una cookie en el ordenador del cliente usamos la función **setcookie()**.
- Ésta puede recibir hasta seis argumentos, aunque sólo el primero es obligatorio, siendo los demás opcionales.
- Los seis parámetros que se le pueden pasar a esta función aparecen en la siguiente tabla.

2. COOKIES

Argumento	Dato	Uso
Nombre de la variable	Cadena	Este parámetro es obligatorio y se usará como índice de la matriz asociativa \$_COOKIE para recuperar el contenido.
Valor de la variable	Cadena Numérico Booleano	El valor de la variable. Si no se pone, la función elimina la cookie en el equipo del cliente.
Duración	Numérico	Se usa para determinar el tiempo durante el que la cookie es válida. Si se intenta leer la cookie después del plazo establecido, no se puede y hay que grabarla de nuevo. Si no se incluye este parámetro, la cookie expira al cerrar el navegador.
Dominio	Cadena	El dominio desde el que se leerá la cookie. Si no se incluye, se puede leer desde cualquier dominio.
Ruta de acceso	Cadena	La carpeta, dentro del servidor, desde la que se leerá la cookie. Si no se incluye, se puede leer desde cualquier carpeta.
Página segura	Booleana	Si esta variable vale true, sólo se puede leer la cookie desde una página segura (https).

2. COOKIES

- Para aclarar cómo funciona todo esto vamos a ver unos listados que responden al esquema visto anteriormente.
 - Empezaremos por el que corresponde a la página principal, donde se comprueba si ya existe la cookie.
 - En este ejemplo lo hemos llamado `uso_cookies.php`.
 - En un sitio real este código estaría incluido en `index.php`, que es el nombre que debe tener la página cero de cualquier sitio en Internet.
-

2. COOKIES

```
<?php
    // Se comprueba si existe la cookie.
    if (!isset($_COOKIE["idiomaUsuario"])) {
        // Si no existe, se determina como página la destinada a elegir el idioma.
        $pagina = "pedirIdioma.htm";
    } elseif ($_COOKIE["idiomaUsuario"] == "sp") {
        // Si existe la cookie y el valor de la variable es "sp" se irá a la página en español.
        $pagina = "spanish.htm";
    } else {
        // Si el valor no es "sp" se irá a la página en inglés.
        $pagina = "english.htm";
    }
?>

<html>
    <head>
        <script language="javascript" type="text/javascript">
            // Se redirige a la página seleccionada.
            location.href = "<?php echo $pagina; ?>";
        </script>
    </head>
</html>
```

2. COOKIES

- Observa cómo se intenta leer el valor de la variable `$_COOKIE["idiomaUsuario"]`.
 - Si la cookie no existe, se fija el nombre de la página a la que saltaremos como la destinada a preguntarle al usuario su idioma. Si existe la cookie y el valor de la variable es `sp`, se irá a la página en español. Si no, se irá a la página en inglés.
 - Hemos diseñado este ejemplo de modo que sólo incluya dos posibilidades de idioma, para simplificarlo, pero podría haber más.
-

2. COOKIES

- El listado de la página que le pregunta el idioma al usuario es muy simple. Lo he llamado pedir_idioma.php:

```
<html>
  <head>
    <title>Pedir el idioma del usuario</title>
    <script language="javascript" type="text/javascript">
      function enviarIdioma (idioma) {
        location.href = "grabar_cookie.php?idiomaUsuario=" + idioma;
      }
    </script>
  </head>

  <body>
    <a href="javascript:enviarIdioma('sp');">Ver la version del sitio en español</a>
    <br>
    <a href="javascript:enviarIdioma('en');">View the english version of the site</a>
  </body>
</html>
```

2. COOKIES

- Como ves, sólo se trata de dos enlaces convencionales.
- Ambos llaman a la misma función JavaScript, cada uno con un argumento que será el valor que represente al idioma elegido.
- La función sólo toma ese argumento y se lo entrega al script encargado de grabar la cookie en el equipo cliente, cuyo nombre es grabar_cookie.php y cuyo listado aparece a continuación.

2. COOKIES

- Observa la línea que graba la cookie.
- El primer parámetro establece el nombre de la variable.
- El segundo establece el valor que tendrá la variable. Este valor se obtiene desde la URL, tal como es invocado este script desde pedir_idioma.php.
- El tercer valor es la fecha de caducidad de la cookie.

```
<?php
    $idiomaUsuario = $_GET["idiomaUsuario"];
    // Se graba una cookie con una validez de 24 horas.
    setcookie("idiomaUsuario", $idiomaUsuario,time()+86400);
?>

<html>
    <body>
        <script language="javascript" type="text/javascript">
            // Se regresa a la página principal.
            location.href = "uso_cookies.php";
        </script>
    </body>
</html>
```

2. COOKIES

- Se ha tomado el momento actual (cuando se ejecuta el script) y se le han sumado 24 horas (86.400 segundos).
 - Si el usuario vuelve a visitar el sitio antes de que hayan transcurrido las veinticuatro horas, se le redireccionará a la página en su idioma.
 - Si vuelve a entrar pasado el plazo establecido (o desde otro ordenador), se le volverá a preguntar su idioma.
-

2. COOKIES

- Normalmente se establece un plazo pensando en la posible frecuencia de visitas.
 - Si no se hubiese especificado el plazo de caducidad, la cookie expiraría al cerrar el navegador, con lo que no tendría mucho sentido en este ejemplo.
 - Si ahora se carga en tu navegador el script `uso_cookies.php`, como es la primera vez que lo cargas y tu cookie aún no existe, se te redireccionará a la página donde se te pregunta tu idioma.
-

2. COOKIES

- Si se pulsa en el enlace correspondiente a la versión en español, eso te llevará a la página encargada de grabar la cookie. Desde ahí saltará a la página principal y, como ya existe la cookie, irá a la página en español.
 - Si ahora cierras tu navegador y, a continuación, lo vuelves a abrir cargando el script `uso_cookies.php` te reconducirá, directamente, a la página en el idioma elegido.
 - Durante 24 horas será así. Pasado el plazo, volverás a ver la selección de idioma.
-

2. COOKIES

- Si quieres eliminar la cookie antes de que se cumpla el plazo, deberás ejecutar el script `borrar_cookie.php`, cuyo listado aparece a continuación:

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

2. COOKIES

- Como ves, es muy simple. En la página no verás nada (queda totalmente en blanco), ya que no hay ningún contenido visualizable.
 - Lo que hacemos es “establecer” la cookie con el parámetro obligatorio del nombre de la variable... y sin ningún otro. Esto elimina la cookie en el cliente. Si ejecutas este script y, a continuación, vuelves a ejecutar `uso_cookies.php` te encontrarás, de nuevo, con la página de selección de idioma.
 - Sin embargo, aún podemos mejorar esto. Cuando se graba una cookie, como en este caso, para que expire a las 24 horas, eso es, exactamente, lo que sucede.
 - Si el usuario entra de nuevo en el plazo establecido, la cookie funciona, pero pasado ese plazo, la cookie expira indefectiblemente, ya que no se renueva de modo automático.
-

2. COOKIES

- Tendríamos que contar con un mecanismo que, al entrar dentro del plazo establecido, renovara la cookie por otro lapso similar.
- Observa el listado del siguiente script, llamado `uso_cookies_renovables.php`:

```
<?php

// Se comprueba si existe la cookie.
if (!isset($_COOKIE["idiomaUsuario"])){
    // Si no existe, se determina como página la destinada a elegir el idioma.
    $pagina = "pedirIdioma.htm";
} elseif ($_COOKIE["idiomaUsuario"] == "sp"){
    /* Si existe la cookie y el valor de la variable es "sp" se irá a la página en español. Además, se vuelve a grabar la cookie por otras 24 horas.*/
    setcookie ("idiomaUsuario", "sp", time()+86400);
    $pagina = "spanish.htm";
} else {
    /* Si el valor no es "sp" se irá a la página en inglés. Además, se vuelve a grabar la cookie por otras 24 horas. */
    setcookie ("idiomaUsuario", "en", time()+86400);
    $pagina = "english.htm";
}

?>

<html>
    <head>
        <script language="javascript" type="text/javascript">
            // Se redirige a la página seleccionada.
            location.href = "<?php echo $pagina; ?>";
        </script>
    </head>
</html>
```

2. COOKIES

- Observa que si la cookie ya existe vuelve a ser grabada de nuevo, prolongando, así, su periodo de validez.
 - Hasta ahora sólo hemos hablado de los tres primeros argumentos.
 - Si establecemos el cuarto, que corresponde al nombre de dominio, la cookie sólo podrá ser leída desde el dominio que se especifique en este parámetro. Esto no es aconsejable si piensas que puedes cambiar el nombre de tu dominio en el futuro.
-

2. COOKIES

- Si estableces un directorio en el quinto parámetro, el script que lee la cookie deberá estar en dicho directorio en el servidor, o no se podrá leer la cookie.
 - Por último, si estableces el sexto argumento con el valor true, la cookie sólo podrá ser leída por un script al que se acceda mediante una capa segura (protocolo https://). Esto es importante si la cookie almacena algún valor sensible, tal como un número de tarjeta de crédito o similar (en una tienda virtual, por ejemplo). Grabar datos potencialmente delicados en cookies es muy desaconsejable.
-

2. COOKIES

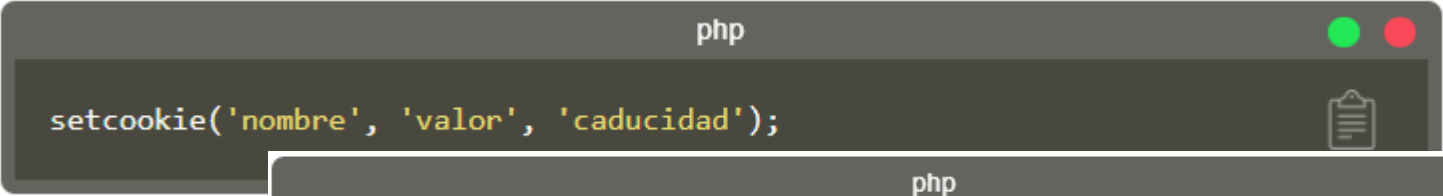
- También podemos grabar cookies que contengan una matriz.
 - Por ejemplo, supongamos el siguiente fragmento de código:
 - `$matriz["primeraClave"] = "Primer elemento";`
 - `$matriz["segundaClave"] = "Segundo elemento";`
 - `$matriz["terceraClave"] = "Tercer elemento";`
 - `setcookie ("matriz", $matriz, time() + 86400);`
 - Con esto habremos grabado una matriz en una cookie. Para recuperar un elemento dado al leer la cookie usaremos lo siguiente:
 - `$primerValor = $_COOKIE["matriz"]["primeraClave"];`
-

2. COOKIES

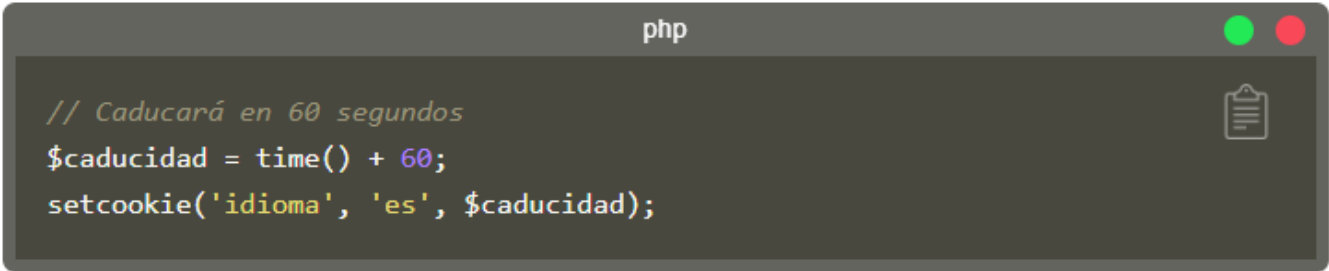
- Una cosa importante. Cuando utilices la función `setcookie()`, debes colocarla en tu script antes de que se produzca ninguna salida hacia la página.
 - Es decir, antes de cualquier sentencia `echo`, `print()`, `printf()` o cualquier dato visualizable.
 - Además, debe ir antes de ninguna etiqueta HTML. Esto se debe a que esta función trabaja a nivel de cabeceras, un tema sobre el que hablaremos en otro artículo.
-

2. COOKIES

- CREACIÓN



```
setcookie('nombre', 'valor', 'caducidad');
```



```
// Caducará en 60 segundos  
$caducidad = time() + 60;  
setcookie('idioma', 'es', $caducidad);
```

- `setcookie(nombre, valor, duración, dominio, ruta, segura);`
 - **Nombre:** Nombre que se le da a la cookie.
 - **Valor:** Valor que le damos a la cookie.
 - **Duración:** Por defecto toma el valor 0, que significa que la cookie caduca en cuanto se cierre el navegador con el que se creó.
 - **Dominio:** Si no se indica nada, toma como dominio en el que se aplica la cookie el dominio actual, pero se puede indicar otro.
 - **Ruta:** Dirección desde la que se lee la cookie.
 - **Segura:** Por defecto vale false. Si indicamos true, entonces indica que la cookie se almacena si estamos comunicándonos mediante protocolo seguro (HTTPS).
-

2. COOKIES

- CREACIÓN

- Ejemplo 1

- Graba la cookie “visitas” con valor 1 y caduca en esta sesión (cuando el usuario cierra el navegador)

- `setcookie("visitas","1") ;`

```
htdocs - ejercicio01.php
1  <?php
2      setcookie("visitas","1");
3  ?>
4
5  <!DOCTYPE html>
6  <html lang="en">
7  <head>
8      <meta charset="UTF-8">
9      <meta http-equiv="X-UA-Compatible" content="IE=edge">
10     <meta name="viewport" content="width=device-width, initial-scale=1.0">
11     <title>Cookies</title>
12 </head>
13 <body>
14
15 </body>
16 </html>
```

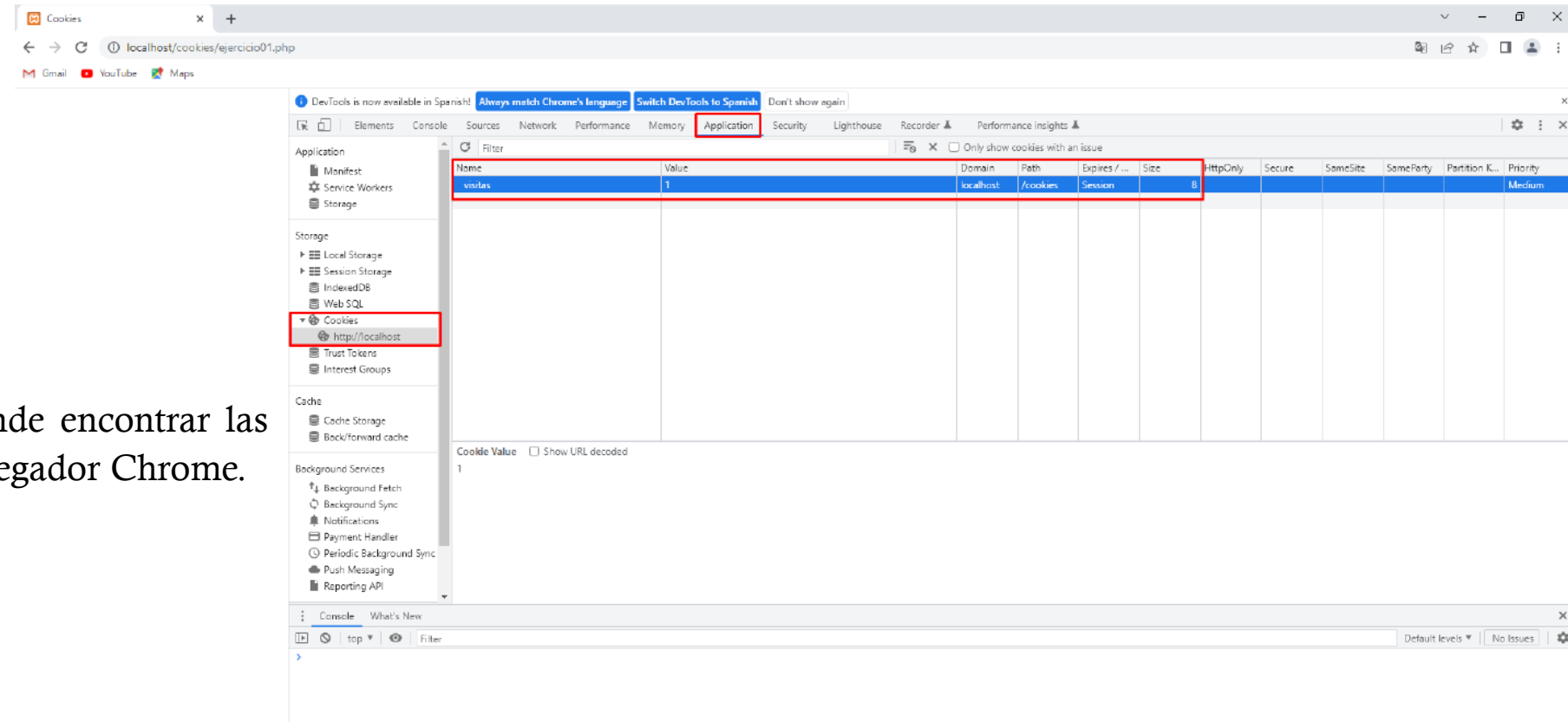
2. COOKIES

- CREACIÓN

- Ejemplo 1

- Visualización

- Vamos a ver dónde encontrar las cookies en el navegador Chrome.



2. COOKIES

- CREACIÓN

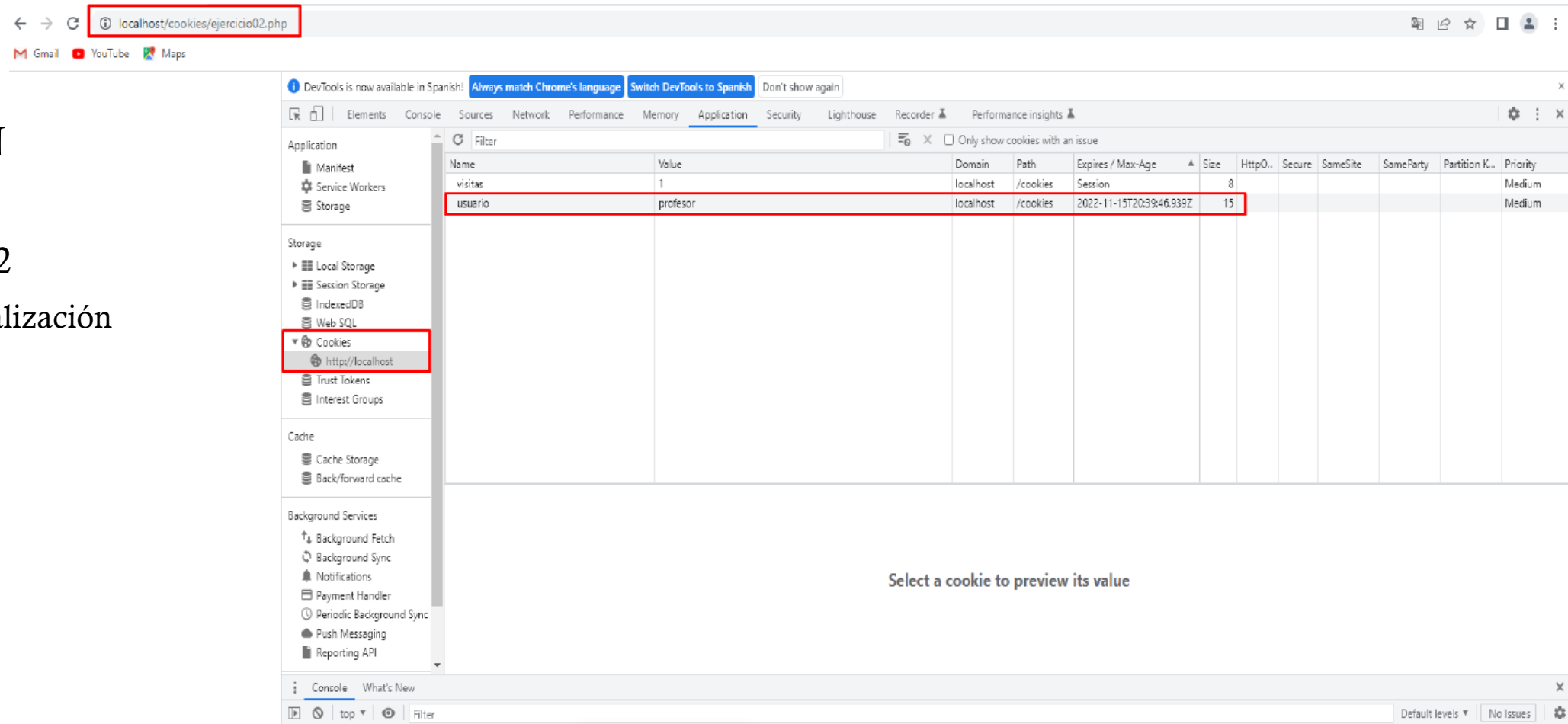
- Ejemplo 2

- Graba la cookie con nombre “usuario” y valor “profesor” y caducará al día siguiente
 - `setcookie("usuario", "profesor", time()+60*60*24);`

```
htdocs - ejercicio02.php
1  <?php
2      setcookie("usuario","profesor",time()+60*60*24);
3  ?>
4  <!DOCTYPE html>
5  <html lang="en">
6  <head>
7      <meta charset="UTF-8">
8      <meta http-equiv="X-UA-Compatible" content="IE=edge">
9      <meta name="viewport" content="width=device-width, initial-scale=1.0">
10     <title>Cookies</title>
11 </head>
12 <body>
13
14 </body>
15 </html>
```

2. COOKIES

- CREACIÓN
 - Ejemplo 2
 - Visualización



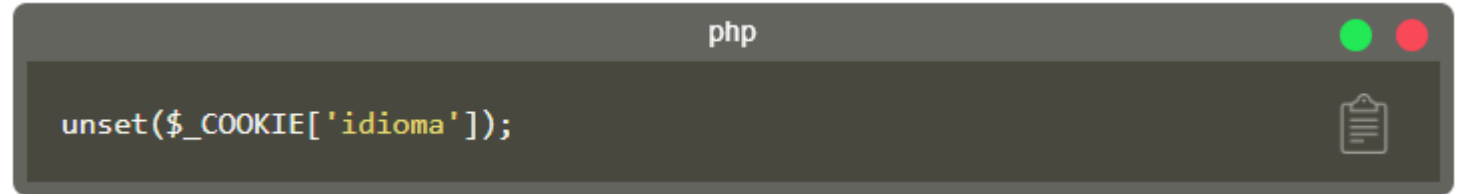
2. COOKIES

- ACTUALIZAR
 - Cuando se vuelve a utilizar la función setcookie indicando el nombre de una cookie ya existente y un nuevo valor, el nuevo valor sobrescribe el anterior valor que tuviera dicha cookie.
 - `setcookie("visitas", 0) ;`

2. COOKIES

- ACCESO
 - Para ello utilizamos el array global `$_COOKIE`, que permite acceder a las cookies almacenadas.
 - `if (isset($_COOKIE["visitas"])){`
 - `echo "Ésta es tu visita número ".$_COOKIE["visitas"];`
 - `}`

2. COOKIES

A code editor window with a dark background and a title bar that says 'php'. The code inside is `unset($_COOKIE['idioma']);`. There are standard window control buttons (green, yellow, red) in the top right corner and a clipboard icon in the bottom right corner.

- BORRAR
 - Para ello utiliza la función `setcookie()` con una fecha de expiración anterior a la actual.
 - Ejemplo: `setcookie("visitas", " ", time()- 60*60*24*7);`
 - Elimina la cookie de nombre visitas con fecha de caducidad de hace al menos una semana
 - También tenemos `unset()`
 - Si usáramos la función de PHP `unset()` únicamente la borraríamos del array asociativo `$_COOKIE` pero continuaría en el navegador web del usuario, y al recargarse la página o ser cargada otra se volvería a recuperar su valor.

2. COOKIES

- Almacenar datos binarios
 - En las cookies no se pueden almacenar datos binarios.
 - Las cookies solo admiten valores de tipo String.
 - Este código no sería válido:
 - `setcookie("notas",array(9,7,6,5)); // inválido: datos binarios`

2. COOKIES

- Almacenar datos binarios
 - Para llevar a cabo esto de manera efectiva, hay que convertir los datos binarios en formato texto.
 - De binario a texto:
 - `setcookie("notas", serialize(array(9,7,6,5)));`
 - De texto a binario:
 - `$array = unserialize($_COOKIE["notas"]);`

2. COOKIES

- Es muy importante que los nombres de las cookies no coincidan con los nombres de controles de formularios, porque PHP incluye los valores de las cookies en `$_REQUEST`.
- Hay que tener la precaución de utilizar la función `setcookie()` antes de empezar a escribir el contenido de la página, porque si no PHP producirá un aviso y no se creará la cookie.

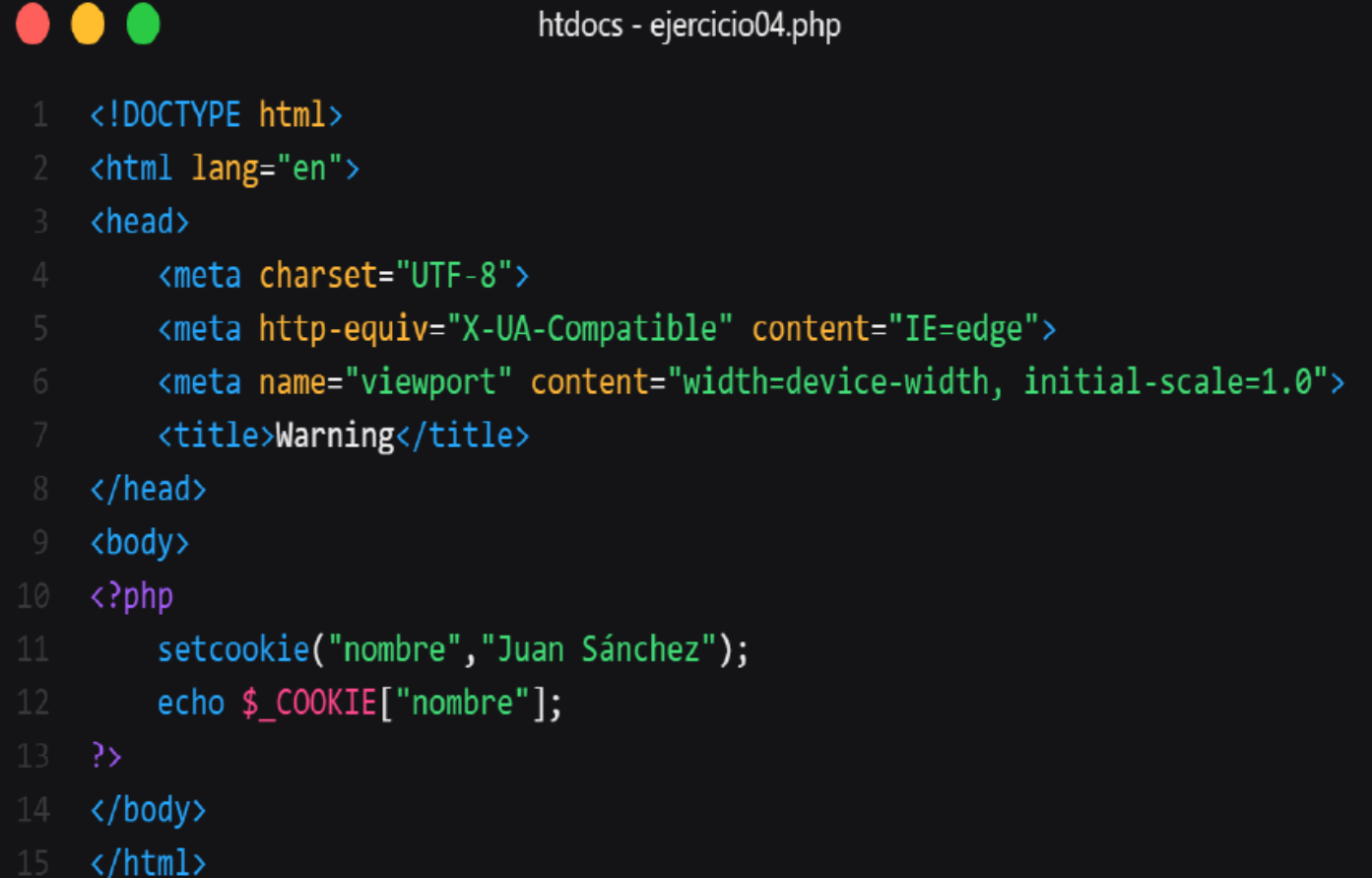
2. COOKIES

- Ejemplo 3
 - Mediante el uso de cookies obtener el número de veces que ha accedido el usuario a la página.

```
htdocs - ejercicio03.php
1  <?php
2      if (isset($_COOKIE["visitas"])){
3          setcookie("visitas",$_COOKIE["visitas"]+1);
4          $mensaje="Ha accedido a la página ".$_COOKIE['visitas']." veces";
5      } else{
6          setcookie("visitas","1");
7          $mensaje = "Bienvenido a la Página";
8      }
9  ?>
10 <!DOCTYPE html>
11 <html lang="en">
12 <head>
13     <meta charset="UTF-8">
14     <meta http-equiv="X-UA-Compatible" content="IE=edge">
15     <meta name="viewport" content="width=device-width, initial-scale=1.0">
16     <title>Cookies</title>
17 </head>
18 <body>
19     <?php
20         echo $mensaje;
21     ?>
22 </body>
23 </html>
```

2. COOKIES

- Ejemplo 4
 - Generando un Warning porque se está mandando texto antes que realizar la creación de la cookie.



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7     <title>Warning</title>
8 </head>
9 <body>
10 <?php
11     setcookie("nombre","Juan Sánchez");
12     echo $_COOKIE["nombre"];
13 ?>
14 </body>
15 </html>
```

2. COOKIES

```
;output_buffering=4096  
output_buffering=0
```

- Ejemplo 4
 - Visualización
 - En un entorno de producción se suele utilizar un buffer de salida, pero puede enmascarar errores de programación. Si no se genera el Warning, vete a php.ini, modifica output_buffering y dale valor 0, reinicia Apache y ejecuta de nuevo.



Warning: Cannot modify header information - headers already sent by (output started at C:\xampp\htdocs\cookies\ejercicio04.php:1) in C:\xampp\htdocs\cookies\ejercicio04.php on line 11
Juan Sánchez

2. COOKIES

- Ejemplo 5
 - Generar una página en la que por defecto el body será de color blanco, si tiene cookie inicializada se le pondrá ese valor y si no la tiene se inicializa al valor que desee el usuario mediante un formulario.

```
htdocs - ejercicio05.php

1  <?php
2      if (isset($_REQUEST['seleccionar'])){
3          setcookie("color",$_REQUEST['dato'], time()+60*60*24);
4          $color = $_REQUEST['dato'];
5      }else if (isset($_COOKIE['color'])){
6          $color = $_COOKIE['color'];
7      }else{
8          $color = "white";
9      }
10  ?>
11  <!DOCTYPE html>
12  <html lang="en">
13  <head>
14      <meta charset="UTF-8">
15      <meta http-equiv="X-UA-Compatible" content="IE=edge">
16      <meta name="viewport" content="width=device-width, initial-scale=1.0">
17      <title>color</title>
18  </head>
19  <body <?php echo "style='background-color:".$color."' ?>>
20      <form method="post" action=<?php $_SERVER['PHP_SELF']?>>
21          Escoge tu color de fondo:<input type="color" name="dato">
22          <input type="submit" name="seleccionar" value="Seleccionar">
23      </form>
24  </body>
25  </html>
```

2. COOKIES

- Ejemplo 6
 - Generar una página que tenga un formulario, recoja todos los valores que pasa el usuario a través de un formulario (nombre, apellidos y edad) y cree una cookie denominada datas con todos esos datos.

```
htdocs - ejercicio06.php

1  <?php
2      if (isset($_REQUEST['enviar'])){
3          $datas = [$_REQUEST['nombre'], $_REQUEST['apellidos'], $_REQUEST['edad']];
4          setcookie("datas", serialize($datas), time()+60*60*24);
5      }else if (isset($_COOKIE['datas'])){
6          echo "<pre>";
7          print_r(unserialize($_COOKIE['datas']));
8          echo "</pre>";
9      }
10  ?>
11  <!DOCTYPE html>
12  <html lang="en">
13  <head>
14      <meta charset="UTF-8">
15      <meta http-equiv="X-UA-Compatible" content="IE=edge">
16      <meta name="viewport" content="width=device-width, initial-scale=1.0">
17      <title>color</title>
18  </head>
19  <body>
20      <form method="post" action= <?php $_SERVER['PHP_SELF']?>>
21          Nombre:<input type="text" name="nombre">
22          Apellidos:<input type="text" name="apellidos">
23          Edad:<input type="number" name="edad">
24          <input type="submit" name="enviar" value="Enviar">
25      </form>
26  </body>
27  </html>
```

ACTIVIDAD 1 – POLÍTICA DE COOKIES

- Muestra un cartel avisando que debe aceptar la política de Cookies con un botón.
- Cuando sea pulsado crea una cookie.
- No vuelvas a mostrar el cartel mientras exista.

ACTIVIDAD 2 – ¿QUÉ IDIOMAS HABLAS?

- El visitante debe tener la posibilidad de cambiar el idioma de la página.
 - Crea un botón para actualizar el idioma.
 - Guarda la selección en una cookie.
 - Muestra el texto adecuado dependiendo de la cookie existente. Si quieres en mitad de la página.
 - ES - Bienvenido
 - EN - Welcome
 - IT - Benvenuto
 - FR - Bienvenue
 - Pro:
 - Guarda además el color del fondo.
 - Usa banderas.
-

3. SESIONES

- La gestión de sesiones es un sistema para almacenar pares **nombre-valor** en el lado del servidor.
 - El uso de sesiones en lugar del de cookies está más definido cuando las variables deben permanecer a disposición de los scripts que emplea el usuario durante la navegación por distintas páginas del mismo sitio.
 - Supongamos que navegas por un sitio en el que todas las páginas (o muchas de ellas) deben estar personalizadas para cada usuario en concreto. Quizás el ejemplo más clásico y relevante sea una tienda en Internet.
-

3. SESIONES

- El usuario navega por las páginas de distintos productos, viendo cuáles le interesan y seleccionando uno aquí y otro allí para su compra.
 - Los productos elegidos deben recordarse de alguna manera de una página a otra, para que, cuando el usuario cierre la cesta le podamos mostrar una relación y el importe total.
 - Dado que HTTP es un protocolo sin estados, es decir, no conserva memoria de una página a otra, como ya sabemos, cada compra individual debe almacenarse en un archivo.
-

3. SESIONES

- Sin embargo, es lógico suponer que el usuario no necesita que el recuerdo de esa compra permanezca disponible para usos sucesivos del sitio, así que no es necesario almacenar una cookie en el ordenador del cliente.
 - También podemos considerar otro ejemplo muy clásico: los típicos portales de contactos, donde cada página aparece personalizada con el nombre de usuario de la persona que entra.
 - Así, cuando visita la ficha de otro usuario o le envía un e-mail, queda constancia de quién ha hecho esa visita o quién firma el mensaje.
-

3. SESIONES

- La forma de almacenar los datos necesarios es mediante unos archivos de servidor que se reconocen porque su nombre empieza con **sess_** seguido del número de identificación de la sesión, también llamado número de sesión o **session id**.
 - El session id se genera de forma aleatoria para cada conexión de un usuario al sitio.
 - Es una cadena formada por un elevado número de dígitos alfanuméricos, de modo que cada usuario, al conectarse, crea un nuevo archivo de sesión, único para él, para esa sesión de trabajo.
-

3. SESIONES

- Cuando abandona el sitio (dirigiéndose a otro, o cerrando el navegador), el archivo ya es innecesario.
- Si el usuario se vuelve a conectar a nuestro sitio se creará otra sesión.
- Para que un script pueda trabajar con sesiones es necesario que incluya, antes de usar ningún dato relativo a la sesión, la función `session_start()`.

3. SESIONES

- Esta función no recibe argumentos y devuelve un valor booleano true, salvo que se haya producido algún error.
 - Tiene dos misiones.
 - Si es la primera vez que se ejecuta en el sitio para una visita en concreto, crea un nuevo archivo de sesión, con un session id exclusivo para ese uso en concreto.
 - Si ya está creada la sesión, la “abre”, es decir, facilita el acceso a las variables que haya registradas.
-

3. SESIONES

- Para registrar una variable se crea cómo un elemento de una matriz específica de PHP llamada `$_SESSION`.
- Cada uno de estos elementos es una variable de sesión que estará disponible, a través de dicha sesión, para otras páginas del sitio.

3. SESIONES

- Así, siguiendo con el ejemplo de los idiomas, podemos guardar el idioma elegido por el usuario en una sesión, así:
 - `$_SESSION["idioma"] = "es";`
 - Cuando necesitemos recuperar el idioma en otra página del sitio, lo haremos así:
 - `$idioma = $_SESSION["idioma"];`
 - Se puede regrabar y leer una variable de sesión tantas veces cómo quieras, pero recuerda iniciar cada script con `session_start()`;
-

3. SESIONES

- ¡Importante! - Como ya hemos mencionado, la sesión en curso se identifica mediante el session id.
 - Este es un valor que se almacena en una constante del sistema que, por defecto, se llama **PHPSESSID**.
 - El nombre puede ser cambiado en el archivo de configuración de PHP, aunque mi recomendación es que no lo cambie.
-

3. SESIONES

- La función `session_name()`, que no recibe argumentos, nos devuelve el nombre de esta constante.
 - Por otra parte tenemos la función `session_id()`, también sin parámetros, que nos devuelve el valor de la constante de identificación, es decir, el session id de la sesión en curso.
 - Estas dos funciones son muy necesarias ya que, para poder leer las variables almacenadas en un fichero de sesión es necesario que esta constante, con su valor, pase de una página a otra.
 - Para ello, podemos usar un campo oculto de un formulario, que se puede enviar mediante GET o POST, o podemos incluir esta información en la barra de direcciones.
-

3. SESIONES

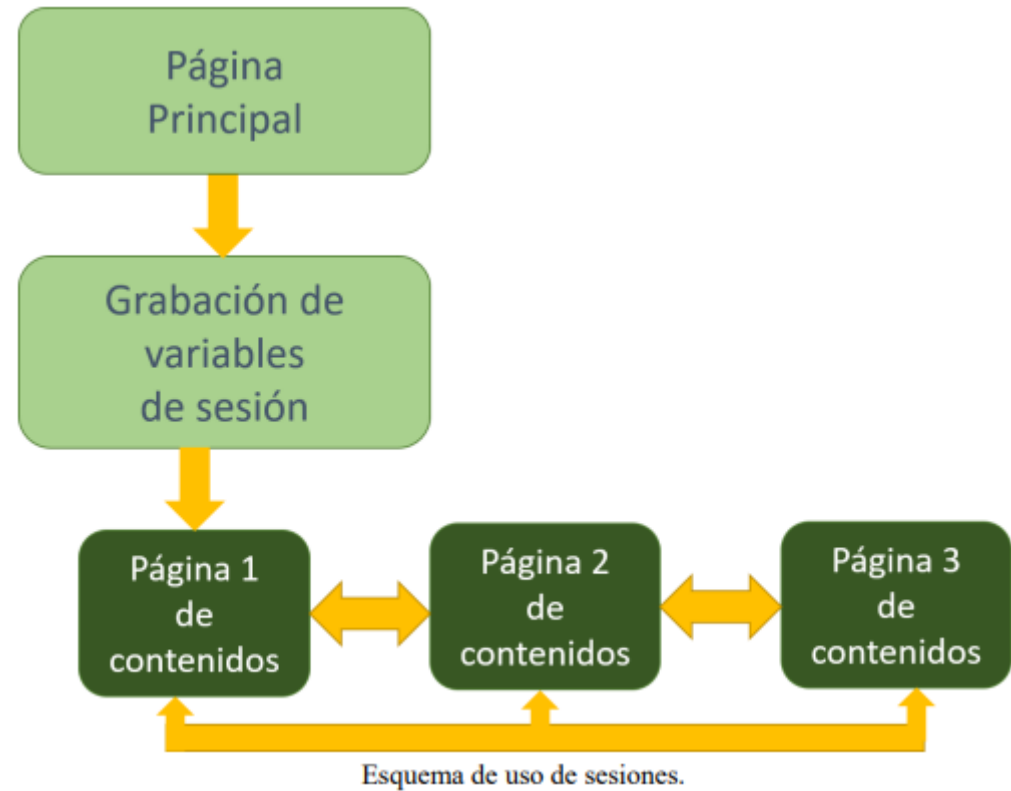
- ATENCIÓN.
 - Existe una forma más cómoda de pasar el identificador de sesión de una página a otra del sitio, de modo implícito y transparente al programador.
 - Se trata de activar la directiva `session.use_trans_sid` en el archivo de configuración.
 - En un artículo posterior aprenderemos lo necesario sobre la configuración de PHP.
-

3. SESIONES

- Aunque hay más funciones que podemos emplear para optimizar la gestión de sesiones, con las que ya tenemos podemos hacer un uso básico, aunque muy eficiente, de la sesión.
- Para comprender su uso hemos creado un sitio ficticio, muy simple, pero que ilustra perfectamente el uso de esta tecnología.

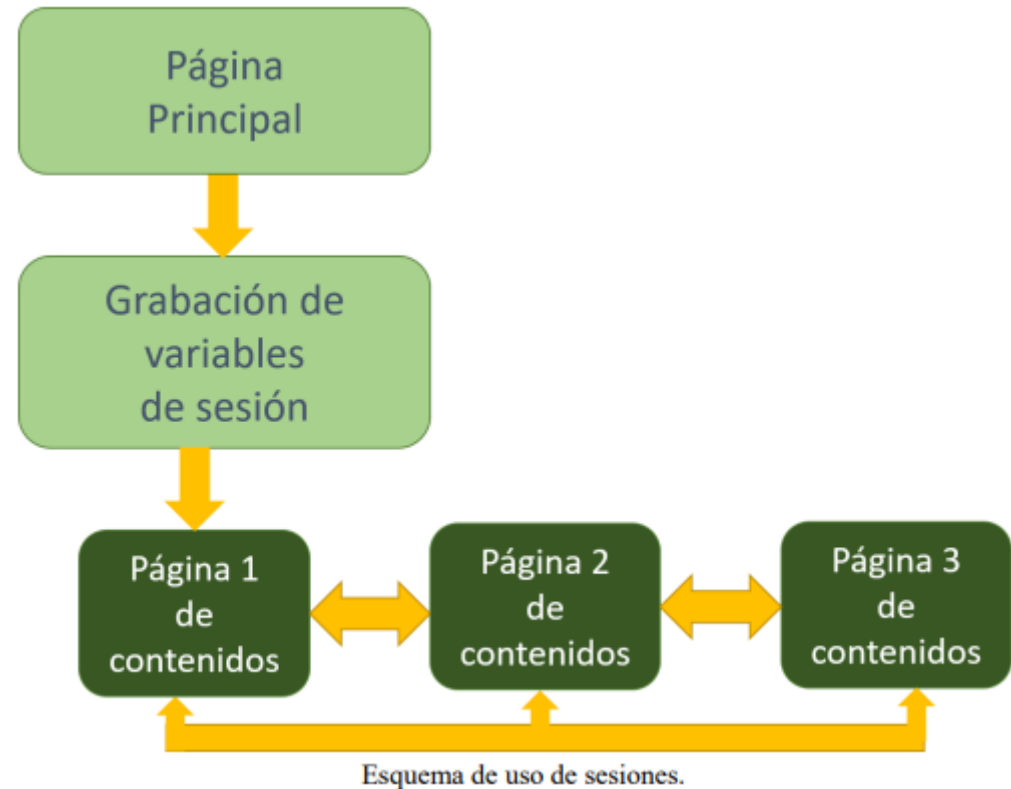
3. SESIONES

- El sitio en cuestión responde al esquema de la imagen siguiente:



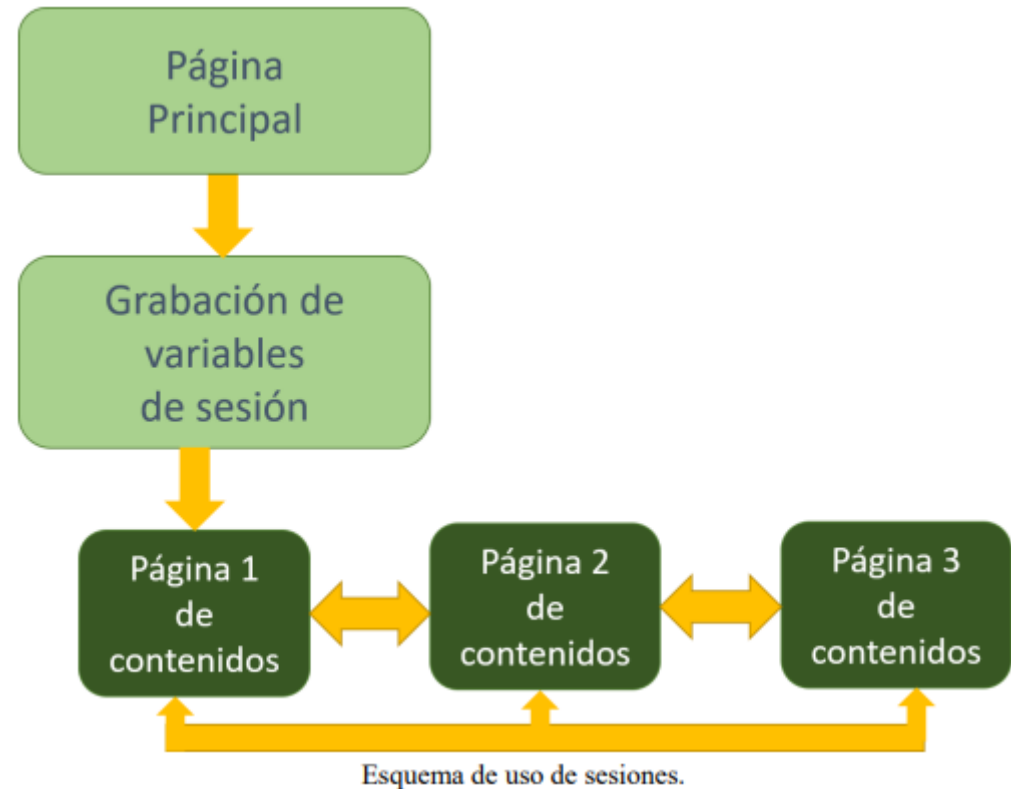
3. SESIONES

- En primer lugar, tenemos una página principal, donde le vamos a pedir al usuario dos datos, a modo de ejemplo, que deberán estar disponibles durante toda la navegación por el sitio.
- Se elige un nombre de usuario y un color para texto.
- Podríamos haber elegido otros datos. Sólo es un ejemplo.



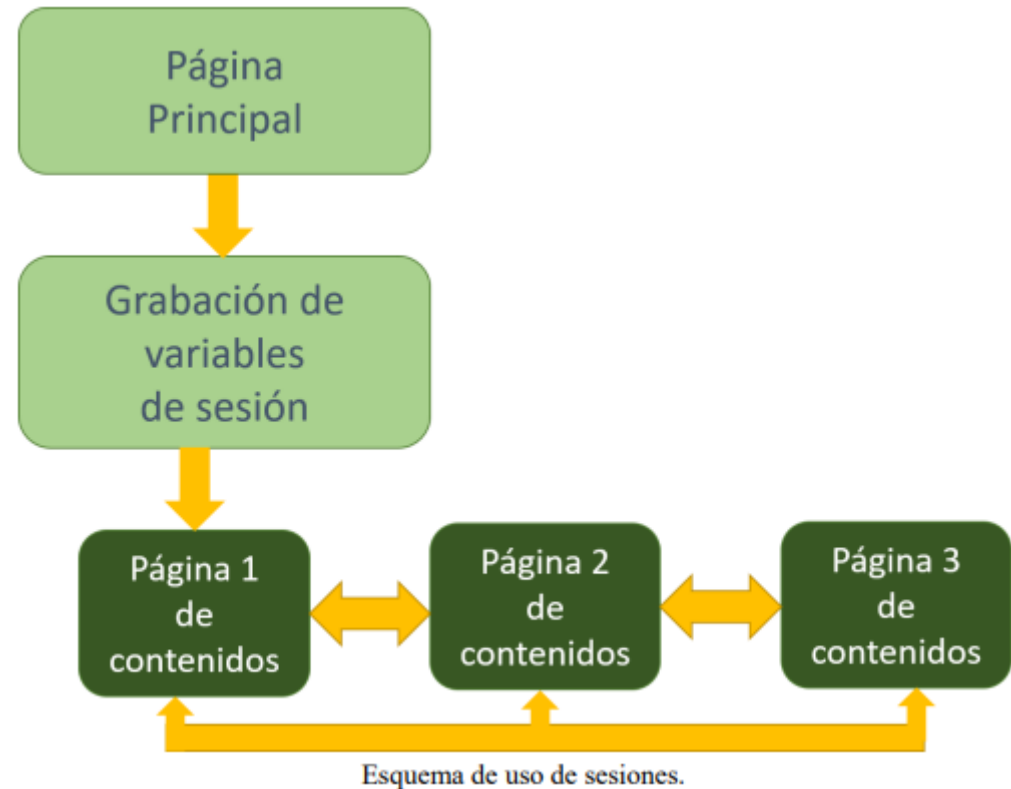
3. SESIONES

- A continuación, tenemos una página donde dichos datos se graban en las correspondientes variables de sesión.
- Esta página es transparente al usuario, de forma que no tiene por qué percibir nada.



3. SESIONES

- Por último, tenemos tres páginas que, en un sitio real, serían de contenidos.
- En ellas hemos hecho que se muestre el nombre del usuario y, en el color elegido, la página en la que está navegando en cada momento.
- Cada una de estas páginas tiene sendos enlaces que permiten el acceso a las otras páginas del sitio.



3. SESIONES

- A continuación, vamos a reproducir los listados de las páginas de este sitio.
- No las cargues en el navegador todavía. Primero veamos los códigos y luego los probaremos.
- El script usado en la página principal se llama, en este ejemplo, **uso_sesiones_inicio.php**, y es el siguiente:

```
<?php
/* Se inicia la sesión. Como no hay ninguna sesión activa,
PHP la se crea en este momento, asignándole, como session_id,
una cadena alfanumérica formada por guarismos aleatorios.
El nombre del fichero de sesión en el servidor sera sess_,
seguido de dicha cadena. */
session_start ();

/* Se indica que variables van a formar parte del archivo de la sesión, aunque,
de momento, no tienen valor alguno asignado. */
$_SESSION["nombre"] = "";
$_SESSION["color"] = "";
?>

<html>
<head>
<title>Uso de sesiones</title>
</head>
<body>
<!-- Se crea un formulario donde se van a grabar dos datos:
el nombre del usuario y el color que elige de una lista.
Este formulario se envía al script que tiene la misión de
almacenar los valores de estos campos en las variables de sesión.
Este script es transparente al usuario. -->
<form method="post" action="pagina_cero_sesiones.php">
<!--Nombre de usuario:-->
<input type="text" name="nombreDeUsuario" id="nombreDeUsuario" value="">
<br>
<!--Color:-->
<select name="colorElegido" id="colorElegido">
<option value="#FF0000">Rojo</option>
<option value="#00FF00">Verde</option>
<option value="#0000FF">Azul</option>
</select>
<br>
</form>
</body>
</html>
```

3. SESIONES

- Al principio del documento ves el uso de la función `session_start()` que hemos comentado anteriormente.
- Observa que no tenemos ningún campo para pasar el identificador de sesión, porque yo ya tengo activado el paso implícito de este parámetro.

3. SESIONES

- El formulario carga contra el script cuyo código aparece a continuación y cuyo nombre es pagina_cero_sesiones.php:

```
<?php
// Se define el salto de línea
define ("salto","<br>n");
session_start();

/* Los dos campos del formulario son asignados a las variables de sesión. */
$_SESSION["nombre"] = $_POST["nombreDeUsuario"];
$_SESSION["color"] = $_POST["colorElegido"];
?>

<html>
<head>
  <script language="javascript" type="text/javascript">
    /* La siguiente función, que se ejecuta a la carga de la página, llama a la primera
    página de contenidos, pasándole, como campo oculto, la constante de id. de la sesión. */
    function mandar(){
      location.href = "pagina_uno.php";
    }
  </script>
</head>

  <body onLoad="javascript:mandar();">
  </body>
</html>
```

3. SESIONES

- Como ves, este script sólo toma los datos procedentes del formulario anterior y los asigna a las correspondientes variables de sesión.
- A continuación, salta a la primera página de contenidos, cuyo nombre es **pagina_uno.php**:

3. SESIONES

- Observa que se reabre la sesión, con **session_start()** y, a partir de ahí, las dos variables que hemos definido como variables de sesión están disponibles.
- Los listados de **pagina_dos.php** y **pagina_tres.php** no los reproduzco aquí por su similitud con el que tienes a la vista.

```
<html>
  <head>
    <title>Uso de sesiones</title>
  </head>

  <body>
    <?php
      // Se define el salto de línea.
      define ("salto","<br>");

      /* Se abre la sesion. */
      session_start();

      /* Se muestra el nombre del usuario y,
      en el color elegido, la página en la que estamos. */
      echo "El nombre de usuario es: " . $_SESSION["nombre"] . salto;
      echo "El color elegido es:
      <h1><fontcolor='" . $_SESSION["color"] . "'>P&Aacute;GINA UNO</font></h1>" . salto;
    ?>

    <a href="pagina_dos.php">Ir a p&aacute;gina 2</a>
    <br>
    <a href="pagina_tres.php">Ir a p&aacute;gina 3</a>
  </body>
</html>
```

3. SESIONES

- Veamos cómo funciona este conjunto de scripts.
 - En primer lugar, carga en el navegador **uso_sesiones_inicio.php**.
 - Verás un formulario con dos campos, en el que se te pide tu nombre de usuario y que elijas un color.
 - Al enviar el formulario mediante el botón habilitado al efecto, estos dos datos viajan al servidor, al script **pagina_cero_sesiones.php**.
-

3. SESIONES

- En este script se asignan las variables del formulario a las correspondientes variables de sesión, tras haber iniciado la sesión con **session_start()**.
 - A partir de ese momento, estas variables quedan disponibles para todas las páginas, mientras dure la visita.
 - El script nos redirecciona automáticamente a **pagina_uno.php**.
-

3. SESIONES

- Los scripts de **pagina_uno.php**, **pagina_dos.php** y **pagina_tres.php** muestran el nombre del usuario y, en el color elegido, muestran el nombre de la página en la que nos encontramos.
 - Como ves, estos datos se obtienen de las correspondientes variables de sesión. Además, cada página posee dos enlaces que te permiten acceder a las demás.
 - La función **session_destroy()** elimina el acceso a la sesión, pero no las variables de la misma, que estarán disponibles de nuevo, si se vuelve a ejecutar **session_start()**.
 - Si queremos eliminar completamente la sesión (por ejemplo, en una página de cierre o desconexión, usaremos la función **session_unset()**.
-

3. SESIONES

- CREACIÓN

```
php
session_start();
$_SESSION['nombre'] = 'Goku';
$_SESSION['raza'] = 'Saiyan';
```

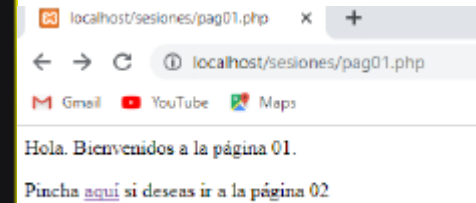
- Las sesiones se crean mediante la función `session_start()`.
 - Si la sesión **no existía**, esta función crea la sesión y le asocia un identificador de sesión único.
 - Si la sesión **ya existía**, esta función permite que la página tenga acceso a la información vinculada a la sesión. Es decir, todas las páginas que quieran guardar datos en `$_SESSION` o leer datos de `$_SESSION` deben comenzar con la función `session_start()`.

```
php
session_start();
echo $_SESSION['nombre'];
// Goku
```

3. SESIONES

- CREACIÓN

```
sesiones - pag01.php
1  <?php
2      session_start();
3      $_SESSION['nombre'] = 'Mariola';
4      $_SESSION['dato'] = 'sesiones';
5      echo "<p>Hola. Bienvenidos a la página 01.</p>";
6      echo "<p>Pincha <a href='./pag02.php'>aquí</a> si deseas ir a la página 02</p>";
7  ?>
```

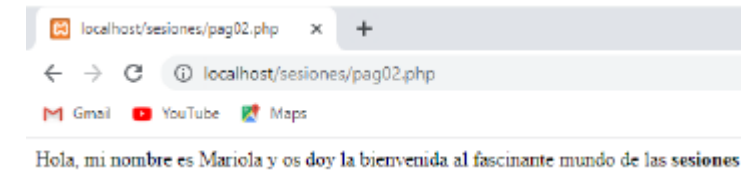


- EJEMPLO

- Vamos a crear dos página (pag01.php y pag02.php) que comparten un nombre y un valor. La primera página crea la sesión, inicializa los datos (con un nombre y un valor) y muestra al usuario un párrafo que le lleva a la página 2.

- La página 2 solo debe mostrar el nombre del usuario y el valor.

```
sesiones - pag02.php
1  <?php
2      session_start();
3      echo "<p>Hola, mi nombre es ". $_SESSION['nombre'] .
4      " y os doy la bienvenida al fascinante mundo de las <strong>". $_SESSION['dato']. "</strong></p>";
5  ?>
```



3. SESIONES

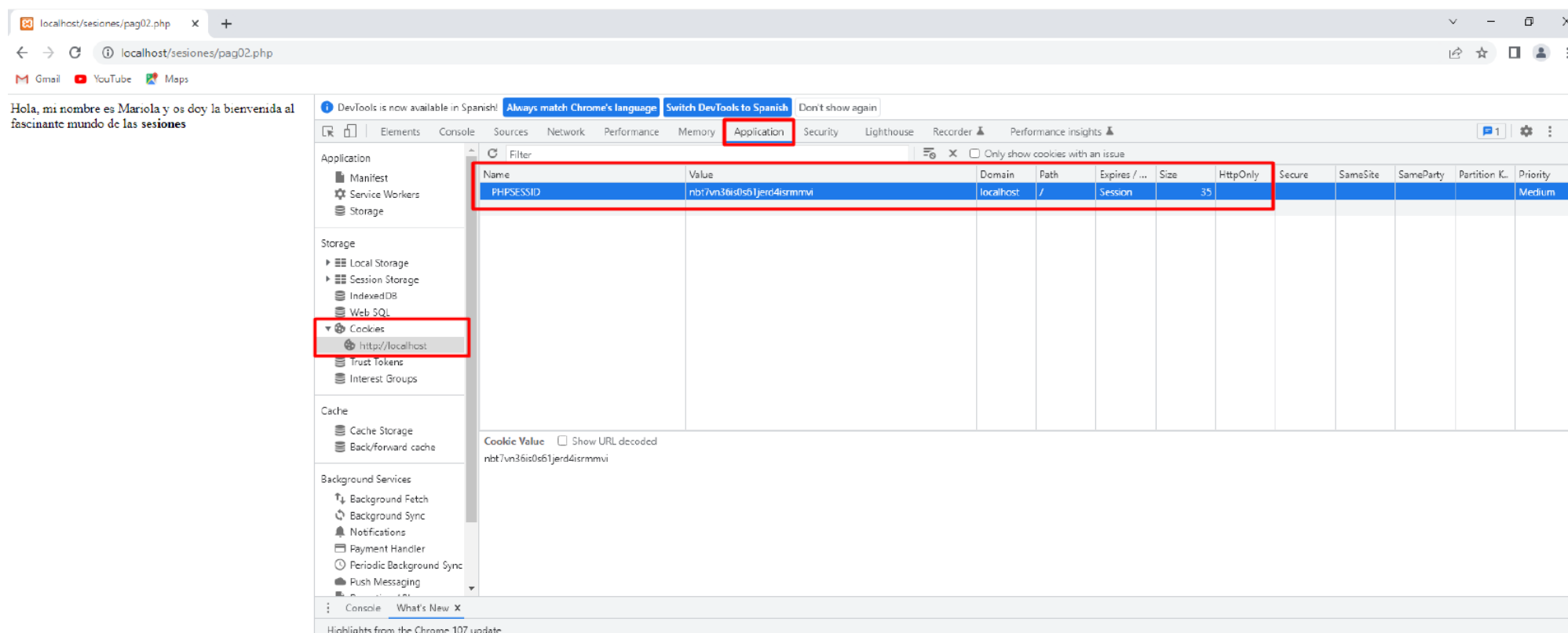
- **CREACIÓN**

- Al crear una sesión, el servidor asigna un identificador al usuario.
 - Este identificador se guarda en el ordenador del cliente en forma de cookie, mientras que en el servidor se guarda en un fichero con los valores de las variables de sesión.
 - Cada vez que el cliente solicita una página al servidor, le envía la cookie que contiene el identificador y así el servidor puede recuperar los valores de la variable de sesión.
 - De esta forma, el cliente no tiene acceso a los valores almacenados en la sesión.
-

3. SESIONES

- CREACIÓN

- EJEMPLO

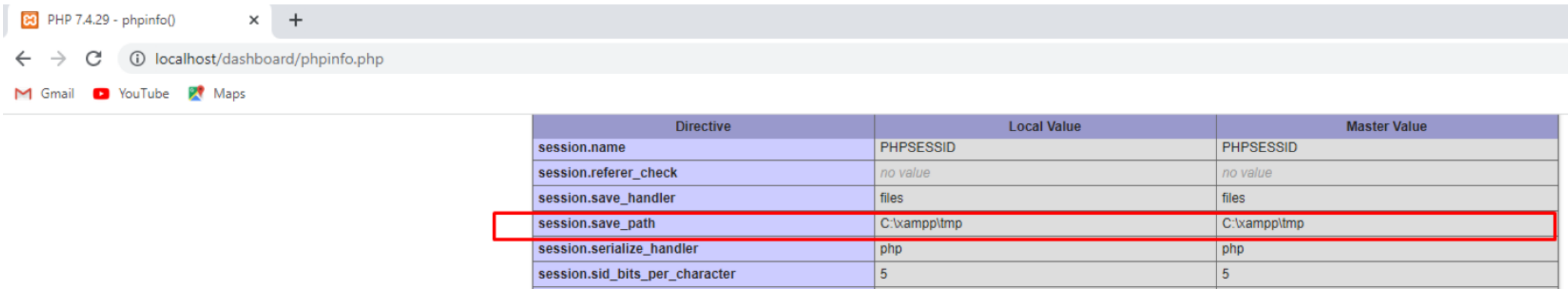


3. SESIONES

- CREACIÓN

- EJEMPLO

- Para saber dónde se almacenan los ficheros de sesión en el servidor, buscamos la directiva de PHP `session.save_path`



The screenshot shows a web browser window with the address bar displaying 'localhost/dashboard/phpinfo.php'. The page content shows the PHP configuration for version 7.4.29. The 'session' section is visible, and the 'session.save_path' directive is highlighted with a red rectangle.

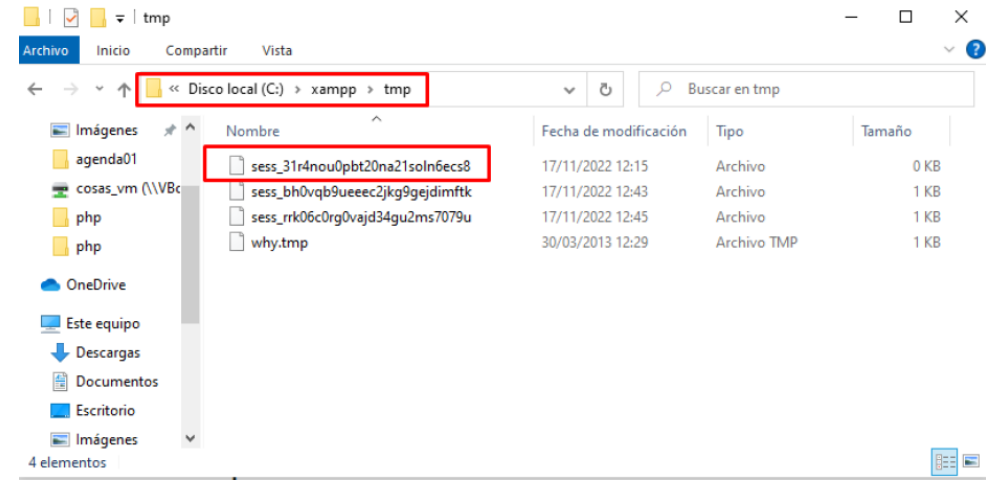
Directive	Local Value	Master Value
session.name	PHPSESSID	PHPSESSID
session.referer_check	no value	no value
session.save_handler	files	files
session.save_path	C:\xampp\tmp	C:\xampp\tmp
session.serialize_handler	php	php
session.sid_bits_per_character	5	5

3. SESIONES

- CREACIÓN

- EJEMPLO

- En ese directorio nos podemos encontrar los distintos ficheros que corresponden a las sesiones (en su nombre viene indicado el SID) – se almacenan en texto plano.



3. SESIONES

- **UTILIZACIÓN**

- Cuando una página ha creado una sesión o ha accedido a una sesión ya existente mediante **session_start()**, la página tiene acceso a la matriz **\$_SESSION** que contiene las variables de esa sesión, siendo un array asociativo accesible desde páginas diferentes (siempre que esas páginas tengan asociada la misma sesión).

3. SESIONES

- UTILIZACIÓN

- EJEMPLO

- Creamos una página denominada ejercicio08.php e insertamos en la variable global `$_SESSION` un nombre directamente, como si de un array numérico se tratara.
 - A continuación, mostramos el array en esta página y creamos un enlace a una página nueva (ejercicio09.php).
 - La nueva página mostrará el array `$_SESSION` y también contendrá un link para poder volver a la página anterior.

```
htdocs - ejercicio08.php

1  <?php
2      session_start();
3      /*Este código no está correcto, puesto que no se
4      puede crear un índice numérico en $_SESSION. En este
5      caso, aunque lo contenga y se muestre, veremos que al
6      llamar a la página 2 el valor no estará disponible*/
7      $_SESSION[] = "Profesor";
8      echo "<p>Valor de \$_SESSION en ejercicio08.php: </p>\n";
9      echo "<pre>"; print_r($_SESSION); echo "</pre>\n";
10     echo "<p>Pincha <a href='./ejercicio09.php'>aquí</a> para ir a la página ejercicio09.php</p>";
11     ?>
```

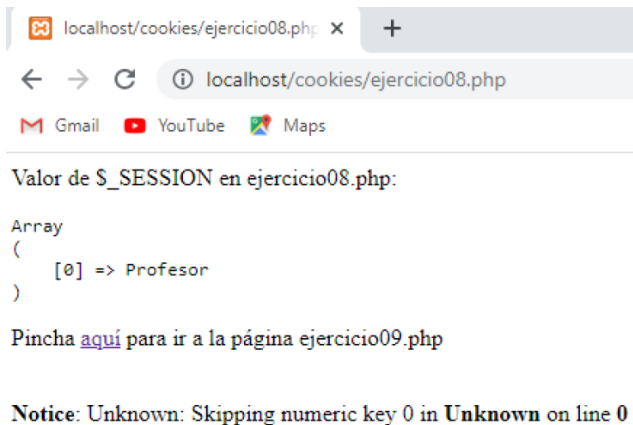
```
htdocs - ejercicio09.php

1  <?php
2      session_start();
3      /*Como puedes ver el array está vacío*/
4      echo "<p>Valor de \$_SESSION en ejercicio09.php: </p>\n";
5      echo "<pre>"; print_r($_SESSION); echo "</pre>\n";
6      echo "<p>Pincha <a href='./ejercicio08.php'>aquí</a> para ir a la página ejercicio08.php</p>";
7      ?>
```

3. SESIONES

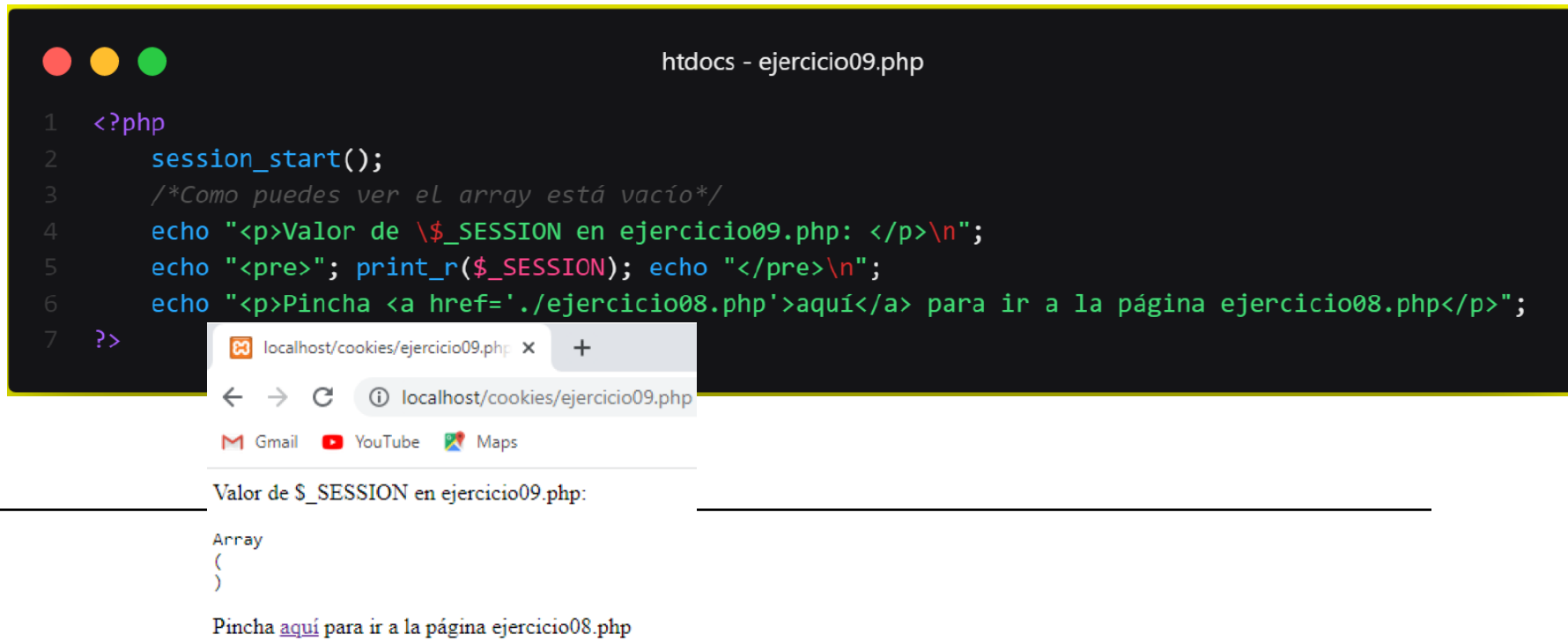
- UTILIZACIÓN

- EJEMPLO



```
htdocs - ejercicio08.php

1  <?php
2      session_start();
3      /*Este código no está correcto, puesto que no se
4      puede crear un índice numérico en $_SESSION. En este
5      caso, aunque lo contenga y se muestre, veremos que al
6      llamar a la página 2 el valor no estará disponible*/
7      $_SESSION[] = "Profesor";
8      echo "<p>Valor de \$_SESSION en ejercicio08.php: </p>\n";
9      echo "<pre>"; print_r($_SESSION); echo "</pre>\n";
10     echo "<p>Pincha <a href='./ejercicio09.php'>aquí</a> para ir a la página ejercicio09.php</p>";
11  ?>
```



3. SESIONES

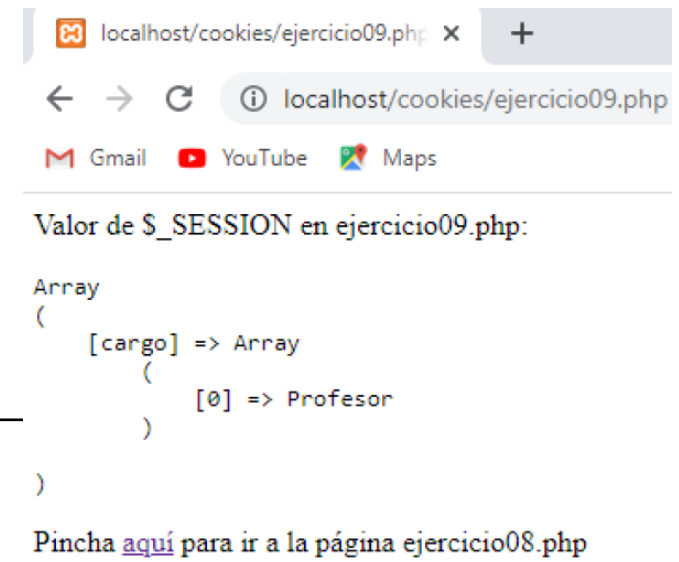
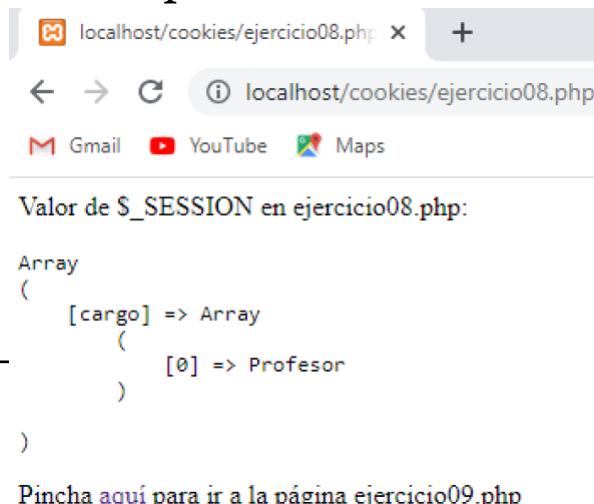
- UTILIZACIÓN

- EJEMPLO

- Creamos un índice que se denomine "cargo" dentro del array asociativo \$_SESSION, en el que dentro iremos poniendo los diferentes elementos en un array numérico.

```
htdocs - ejercicio08.php

1  <?php
2      session_start();
3      /*Este código no está correcto, puesto que no se
4      puede crear un índice numérico en $_SESSION. En este
5      caso, aunque lo contenga y se muestre, veremos que al
6      llamar a la página 2 el valor no estará disponible*/
7      $_SESSION["cargo"][] = "Profesor";
8      echo "<p>Valor de \$_SESSION en ejercicio08.php: </p>\n";
9      echo "<pre>"; print_r($_SESSION); echo "</pre>\n";
10     echo "<p>Pincha <a href='./ejercicio09.php'>aquí</a> para ir a la página ejercicio09.php</p>";
11  ?>
```



3. SESIONES

- **CERRAR**

- Para cerrar la sesión se utiliza la función `session_destroy()`.
- Hay que tener en cuenta que la página que destruye la sesión sigue teniendo acceso a la variable `$_SESSION` y a sus valores, pero las páginas posteriores ya no.

3. SESIONES

- CERRAR

- EJEMPLO

- Vamos a utilizar el ejemplo anterior en el que guardamos nuestro nombre en la variable `$_SESSION` (el fichero se denominará `pag10.php`), destruimos la sesión y a continuación comprobamos si existe la misma.
 - Si existe, muestra el contenido y, si no, que indique un mensaje como: “el nombre no existe”.
 - Además, ponemos un enlace a una página denominada `pag11.php` que también intentará mostrar el contenido si existe y, si no, muestra “el nombre no existe”.

```
htdocs - pag10.php

1  <?php
2      session_start();
3      $_SESSION['nombre']= 'Profesor';
4      session_destroy();
5
6      if (isset($_SESSION['nombre'])){
7          echo "<p>Su nombre es: </p>".$_SESSION['nombre'];
8      }
9      else{
10         echo "Su nombre no existe";
11     }
12     echo "<p>Pincha<a href='./pag11.php'> aquí </a>para ir a la pag11.php</p>";
13     ?>
```

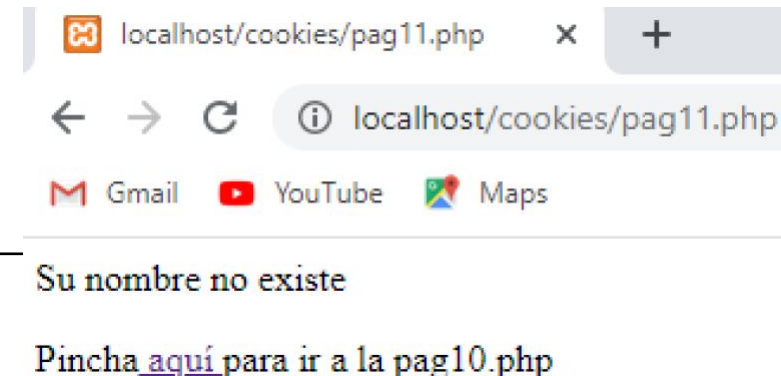
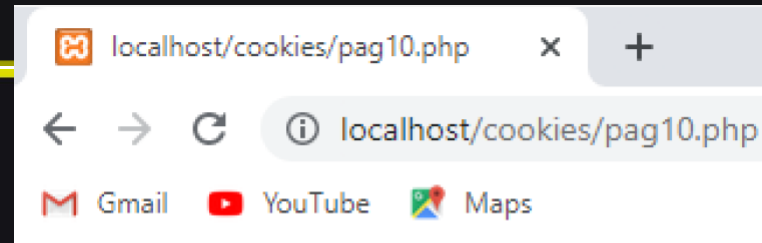
3. SESIONES

- CERRAR

- EJEMPLO

```
htdocs - pag10.php
1  <?php
2      session_start();
3      $_SESSION['nombre']= 'Profesor';
4      session_destroy();
5
6      if (isset($_SESSION['nombre'])){
7          echo "<p>Su nombre es: </p>".$_SESSION['nombre'];
8      }
9      else{
10         echo "Su nombre no existe";
11     }
12     echo "<p>Pincha<a href='./pag11.php'> aquí </a>para ir a la pag11.php</p>";
13     ?>
```

```
htdocs - pag11.php
1  <?php
2      session_start();
3      if (isset($_SESSION['nombre'])){
4          echo "<p>Su nombre es: </p>".$_SESSION['nombre'];
5      }
6      else{
7          echo "Su nombre no existe";
8      }
9      echo "<p>Pincha<a href='./pag10.php'> aquí </a>para ir a la pag10.php</p>";
10     ?>
```



3. SESIONES

- **NOMBRES DE SESIÓN**

- La función utilizada para ello es **session_name()**.
 - Básicamente nos permite establecer nombres de sesión específicos, de forma que todas las páginas que tengan el mismo nombre de sesión accederán a su propia \$_SESSION.
 - Recuerda → Todas las páginas deben tener ese nombre de sesión.
 - Si no lo utilizamos, todas las páginas acceden a la misma \$_SESSION dentro del mismo dominio, puesto que la sesión es única.
-

3. SESIONES

- **NOMBRES DE SESIÓN**

- EJEMPLO

- Crea una página denominada pag12.php en la que el nombre de sesión será pagina12. Crea otra página denominada pag13.php en la que el nombre de sesión será pagina13.
 - A continuación, en la pag12.php guardamos dentro del array asociativo, en el índice “nombre” tu nombre y en la otra página el nombre de otro compañero/a de clase.

3. SESIONES

- **NOMBRES DE SESIÓN**

- EJEMPLO

- Y muestra el siguiente mensaje en las dos páginas: “El nombre es: //Aquí irá el nombre que esté en `$_SESSION`”.
 - Poner un link a la `pag14.php` en `pag12.php` y otro en la `pag13.php` para acceder a `pag15.php`. Ahora crea la `pag14.php` (nombre de sesión `pagina12`) y la `pag15.php` (nombre de sesión `pagina13`) y muestra en cada una de ellas el contenido de `$_SESSION`.
-

3. SESIONES

- NOMBRES DE SESIÓN

- EJEMPLO

```
htdocs - pag13.php
1  <?php
2      session_name("pagina13");
3      session_start();
4      $_SESSION['nombre']="Alumno";
5      echo "<p>El nombre es: </p>".$_SESSION['nombre'];
6      echo "<p>Pincha <a href='./pag15.php'>aquí</a> para ir a la página 15</p>"
7  ?>
```

```
htdocs - pag14.php
1  <?php
2      session_name("pagina12");
3      session_start();
4      echo "<p>El nombre es: </p>".$_SESSION['nombre'];
5      echo "<p>Pincha <a href='./pag12.php'>aquí</a> para ir a la página 12</p>"
6  ?>
```

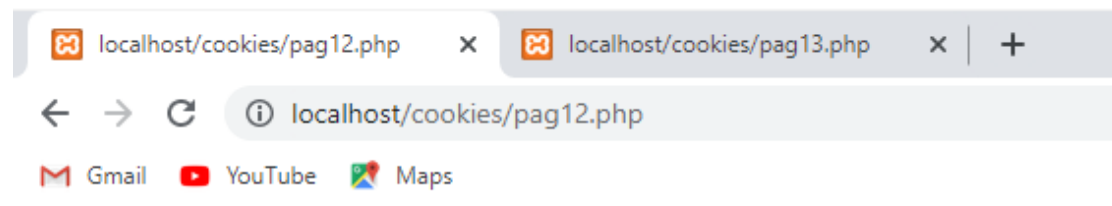
```
htdocs - pag12.php
1  <?php
2      session_name("pagina12");
3      session_start();
4      $_SESSION['nombre']="Profesor";
5      echo "<p>El nombre es: </p>".$_SESSION['nombre'];
6      echo "<p>Pincha <a href='./pag14.php'>aquí</a> para ir a la página 14</p>"
7  ?>
```

```
htdocs - pag15.php
1  <?php
2      session_name("pagina13");
3      session_start();
4      echo "<p>El nombre es: </p>".$_SESSION['nombre'];
5      echo "<p>Pincha <a href='./pag13.php'>aquí</a> para ir a la página 13</p>"
6  ?>
```

3. SESIONES

- NOMBRES DE SESIÓN

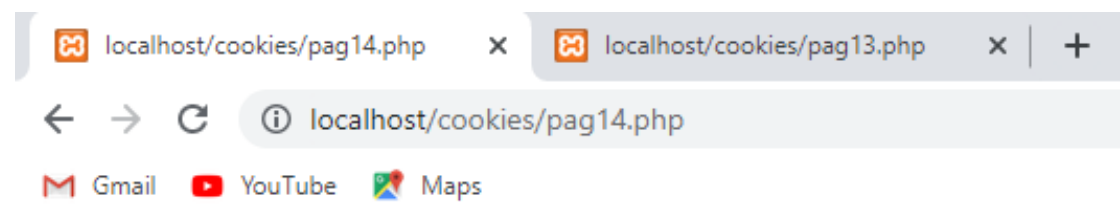
- EJEMPLO



El nombre es:

Profesor

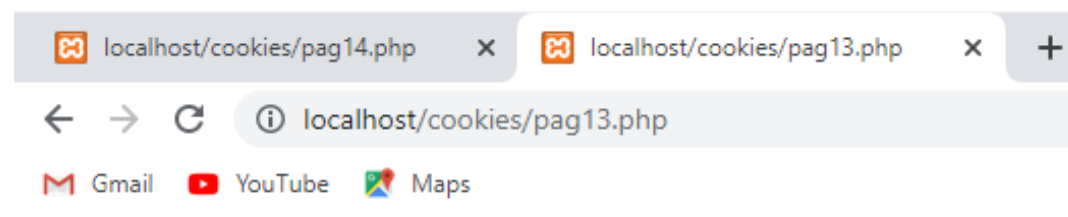
Pincha [aquí](#) para ir a la página 14



El nombre es:

Profesor

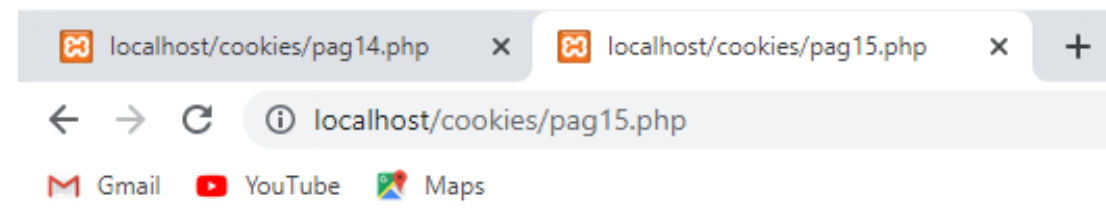
Pincha [aquí](#) para ir a la página 12



El nombre es:

Alumno

Pincha [aquí](#) para ir a la página 15



El nombre es:

Alumno

Pincha [aquí](#) para ir a la página 13

3. SESIONES

- **EJEMPLO**

- Sencillo login con página protegida
 - Vamos a disponer de 2 páginas: una pública y otra privada.
 - **login.php** es pública. Cualquier persona podrá entrar. El objetivo es la que un usuario se identifique y automáticamente le redirija a su página de perfil. Los datos correctos para el formulario son:
 - Apodo: bulma
 - Contraseña: 123
-

3. SESIONES

- EJEMPLO

```
php
<html>
  <body>
    <?php
      // Comprobamos que nos llega los datos del formulario
      if ($_SERVER['REQUEST_METHOD'] == 'POST') {

        // Variables que teóricamente estarían en una base de datos
        $apodoBueno = 'bulma';
        $contraseniaBuena = '123';

        // Variables del formulario
        $apodo = isset($_REQUEST['apodo']) ? $_REQUEST['apodo'] : null;
        $contrasenia = isset($_REQUEST['contrasenia']) ? $_REQUEST['contrasenia'] : null;

        // Comprobamos si los datos son correctos
        if ($apodoBueno == $apodo && $contraseniaBuena == $contrasenia) {
          // Si son correctos, creamos la sesión
          session_start();
          $_SESSION['apodo'] = $_REQUEST['apodo'];
          // Redireccionamos a la página segura
          header('Location: perfil.php');
          die();
        } else {
          // Si no son correctos, informamos al usuario
          echo '<p style="color: red">El apodo o la contraseña es incorrecta.</p>';
        }
      }
    ?>
  </body>
</html>
```

3. SESIONES

- EJEMPLO

```
<form method="post">
  <p>
    <input type="text" name="apodo" placeholder="Apodo">
  </p>
  <p>
    <input type="password" name="contrasenia" placeholder="Contraseña">
  </p>
  <p>
    <input type="submit" value="Entrar">
  </p>
</form>
</body>
</html>
```

```
php
<html>
  <body>
    <?php
      // Comprobamos que nos llega los datos del formulario
      if ($_SERVER['REQUEST_METHOD'] == 'POST') {

        // Variables que teóricamente estarían en una base de datos
        $apodoBueno = 'bulma';
        $contraseniaBuena = '123';

        // Variables del formulario
        $apodo = isset($_REQUEST['apodo']) ? $_REQUEST['apodo'] : null;
        $contrasenia = isset($_REQUEST['contrasenia']) ? $_REQUEST['contrasenia'] : null;
```

```
        if ($apodo == $apodoBueno && $contrasenia == $contraseniaBuena) {
```

```
        } else {
```

```
          // Usuario incorrecto
```

```
          echo "La contraseña es incorrecta.</p>";
```

3. SESIONES

- **EJEMPLO**

- perfil.php es una página que nadie puede entrar si no pasa por la página anterior.
- RECOMENDACIÓN: Cuando hagas una redirección con header() siempre termina con un die() o exit().

3. SESIONES

- EJEMPLO

```
php
<?php
// Comprobamos si existe la sesión de apodo
session_start();
if (!isset($_SESSION['apodo'])) {
    // En caso contrario devolvemos a la página login.php
    header('Location: login.php');
    die();
}
?>
<html>
    <body>
        <!-- Saludamos -->
        <h1>Bienvenido <?= $_SESSION['apodo'] ?></h1>
        <!-- Botón para cerrar la sesión -->
        <a href="logout.php">Cerrar sesión</a>
    </body>
</html>
```

3. SESIONES

- **EJEMPLO**

- logout.php: Para complementar se puede crear una página para cerrar la sesión.



```
<?php
// Iniciamos las sesiones
session_start();
// Destruimos las sesiones
session_destroy();
// Llevamos a login.php
header('Location: login.php');
die();
```

ACTIVIDAD 3 - IDENTIFÍCATE

- Crea una web donde se pida un usuario y una contraseña.
- Si es correcto debe llegar a una página protegida por sesión.
- Añade un botón para cerrar la sesión.