¿Qué son las sesiones?

Suponemos que habrás estado alguna vez en un hotel y que recuerdas que al inscribirnos como huéspedes nos facilitan una *tarjetita identificativa* que teóricamente habríamos de presentar a la hora de solicitar cualquier servicio del hotel (cafetería, restaurante, etc.).

Al registrarnos en ese hotel estaremos **iniciando una sesión** (nuestro período de estancia) y al recibir la tarjeta identificativa se nos estará facilitando **un identificador de sesión**, que tiene validez **temporal**, ya que **expirará** en la *fecha de salida* indicada en ella.

Imaginemos ahora que vamos al *restaurante*.

Pueden ocurrir dos cosas: que decidamos *efectuar el pago directamente*, o que pidamos que el importe de la factura se incluya en *nuestra cuenta*. En el segundo de los casos, se **reiniciará la sesión** y se registrará una nueva **variable de sesión** –el importe del servicio– al firmar la *nota de cargo* del mismo.

El responsable del restaurante guardará esa *nota de cargo* –**una variable de sesión**– de forma **temporal** ya que una vez abonado su importe, en el momento que abandonemos el hotel, expirará la sesión y dejará de tener validez.

Se requiere –por tanto– un **directorio temporal** en el que *almacenar* las **variables de sesión**.

En PHP la sesiones funcionan de forma muy similar y de esa similitud surge la necesidad de habilitar un directorio temporal antes de utilizar *sesiones*. Lo hacemos cuando modificamos la configuración tal como comentamos en la columna de la derecha.

Funciones de sesión

Para la gestión de **sesiones** se utilizan estas funciones:

session_start()

Crea una sesión o continúa con la actual. En el segundo caso el identificador de sesión debe ser *tranferido* por medio de una variable GET o a través de una *cookie*.

session_name()

Recoge el nombre de la sesión.

Si no se asigna uno de forma explícita, utiliza como nombre de sesión el contenido de la directiva **session.name** del fichero **php.ini**.

Por defecto ese nombre suele ser **PHPSESSID**, pero no está de más comprobarlo mirando el valor actual de la directiva **session.name** en info.php.

session_name(*'nombre'*)

Esta función permite asignar un **nuevo nombre** a la sesión actual.

Debemos tener en cuenta que si

Requisitos de configuración

Antes de poder utilizar este tipo de funciones es preciso modificar el fichero de configuración de php. Editemos el fichero **php.ini** y busquemos esta línea:

session.save_path = /tmp

vamos a modificarla. La dejaremos así:

session.save_path="C:\Apache\htdocs\temp"

Al modificar esta configuración hemos asignado *una ruta* y un **directorio** (temp) en el que se guardarán los ficheros temporales creados durante las sesiones, pero... ese *directorio aún no existe*.

Antes de reiniciar el servidor Apache –después de efectuar los cambios anteriores– tendremos que **crear un directorio** con nombre **temp** dentro de **htdocs** y a partir de ese momento ya estaremos en condiciones de reiniciar Apache con PHP habilitado para el uso de sesiones.

Iniciando y propagando sesiones

Aquí tenemos un ejemplo de como *iniciar una sesión* y también podremos comprobar como **se propaga** al llamar a la misma página si está activada la opción **aceptar cookies**.

Si ejecutamos reiteradamente el script –pulsando en *volver a llamar esta página*– podremos ver que no se modifica el valor del **identificador de sesión**.

Si **bloqueamos** *las cookies* (en este enlace está descrito el procedimiento para hacerlo) podremos comprobar que *ahora* la sesión **no se propaga** y que cada vez que *volvemos a llamar a la página* nos aparece un nuevo valor en el **identificador de sesión**, es decir, se crea una nueva sesión.

```
<?
session_start();
#pedimos que escriba el identificador único y el nombre de la sesión
echo session_id(),"<br>";
echo session_name(),"<br>";
?>
<A Href="ejemplo116.php">Volver a llamar estallamar esta página</A>
```

Ejemplo 116.php

En este otro ejemplo solo hemos hecho *el añadido* (señalado en *magenta*) para el caso de que PHP no esté compilado para *--enable-trans-sid* (una opción bastante habitual en los hosting que trabajan bajos sistemas operativos UNIX ó LINUX).

Si lo ejecutamos podremos comprobar que ocurre *exactamente lo mismo* que en el ejemplo anterior, tanto con *cookies* activadas como desactivadas.

```
<?
 session_start();
#pedimos que escriba el identificador único y el nombre de la sesión
 echo session_id(),"<br>";
 echo session_name(),"<br>";
?>
<A Href="ejemplo117.php?<?=$SID ?>">Volver a llamar esta página</A>
```

Ejemplo 117.php

Ahora haremos una **modificación importante** en el script. Al **incluir en la llamada a la página el nombre y el identificador de sesión** estamos transfiriendo –mediante el método GET– el valor de esa variable. De esta forma, la propagación de la sesión estará asegurada sea cual fuere la configuración del navegador del cliente.

Podemos comprobar –activando/desactivando cookies– que en las sucesivas llamadas a la página **se mantiene** el identificador de sesión.

```
<?
session_start();
#pedimos que escriba el identificador único y el nombre de la sesión
echo session_id(),"<br>";
echo session_name(),"<br>";
?>
<A Href="ejemplo118.php?<?echo session_name()."=".session_id()?>">
       Volver a llamar esta página</A>
```

Ejemplo 118.php

Sesiones con nombre propio

En los ejemplos siguientes se crean y propagan *sesiones con nombre propio* sin que ello altere las condiciones de respuesta con las opciones **aceptar /no aceptar cookies**. También hemos incluido **session_cache_limiter** con las restricciones de **nocache** con lo

cambiamos de página y queremos mantener el mismo **identificador** (conservar la sesión anterior) esta función debe ser escrita, con el mismo nombre, en la nueva página, y además, ha de ser *insertada* **antes** de *llamar* a la función **session_start()** para que se inicie la sesión.

session_cache_limiter()

El limitador de caché controla las *cabeceras* HTTP enviadas al cliente.

Estas *cabeceras* determinan las reglas mediante las que se habilita la opción de que los *contenidos* de las páginas puedan ser guardados en la caché local del cliente o se impida tal almacenamiento.

En este último modo –no caché– cada petición de página requeriría una nueva llamada al servidor, lo cual tiene –como todo en la vida– ventajas e inconvenientes

Entre las ventajas está la garantía de que en cada acceso estamos viendo la versión actualizada de la página, cosa que podría no ocurrir de otro modo. El inconveniente es que requiere una nueva petición que puede significar un tiempo de espera, mientras el servidor produce la respuesta.

Esta opción viene configurada *por defecto* en las directivas de configuración del **php.ini** como **nocache**.

Para evitar –sea cual fuera la configuración de php.ini– el almacenamiento de las páginas en la caché del cliente hemos de utilizar:

session_cache_limiter ('nocache,private').

De igual forma que ocurría con **session_name**, si utilizamos está función debemos escribirla **antes** que **session_name** y –también igual que en aquel caso– deberemos repetirla en cada uno de los documentos.

¡Advertencia!

El *orden* de escritura de estas instrucciones sería el siguiente:

```
<?
session_cache_limiter();
session_name('nombre');
session_start();
.....
?>
```

Es muy importante mantener ese orden y que este bloque de instrucciones sea el *primer elemento de la página* -antes de cualquier otra etiqueta- y que *no haya líneas en blanco* ni antes de la etiqueta **<?** ni entre ella y las llamadas a estas funciones.

Propagación de las sesiones

La verdadera utilidad de las sesiones es la posibilidad de que sean propagadas, es decir, que tanto el *identificador de sesión* como los *valores de las variables de sesión* -luego hablaremos de estas variables- puedan ir *pasando* de *una página a otra* sin necesidad de recurrir al uso de formularios.

Para *entendernos*, se trata de *dar validez* y utilizar la misma *tarjeta* para movernos en las diferentes secciones del *hotel* que hemos utilizado como ejemplo.

La forma habitual de *propagar* las

cual el navegador no guardará las páginas de ejemplo en la *caché*.

Si cambiáramos los parámetros (*nocache*, *private* por public) podríamos comprobar se almacena en la caché del navegador la página web devuelta por el servidor.

```
<?
session_cache_limiter('nocache,private');
session_name('leocadia');
session_start();
#pedimos que escriba el identificador único
echo session_id(),"<br>";
echo session_name(),"<br>";
?>
<A Href="ejemplo119.php">Volver a llamar esta página</A>
```

Ejemplo 119.php

```
<?
session_cache_limiter('nocache,private');
session_name('leocadia');
session_start();
#pedimos que escriba el identificador único
echo session_id(),"<br>";
echo session_name(),"<br>";
?>
<A Href="ejemplo120.php?<?=$SID ?>">Volver a llamar esta página</A>
```

Ejemplo 120.php

```
<?
session_cache_limiter('nocache,private');
session_name('leocadia');
session_start();
#pedimos que escriba el identificador único
echo session_id(),"<br>";
echo session_name(),"<br>";
?>
<A Href="ejemplo121.php?<?echo session_name()."=".session_id()?>">
                  Volver a llamar esta página</A>
```

Ejemplo 121.php

Las cookies de sesión y sus parámetros

Para el caso de que el navegador del cliente tenga activada la opción de **aceptar cookies** PHP dispone de una función que permite *leer* los parámetros de esa *cookie*. Para ello dispone de la función **session_get_cookie_params()** que devuelve un **array asociativo** con los siguientes índices:

lifetime: Indica el tiempo de duración de la duración de la cookie
path: Indica la ruta -en el ordenador del cliente- en la que se almacenan los datos de la sesión
domain :Indica el dominio de procedencia de la cookie
secure :Indica si la cookie solo puede ser enviada a través de *conexiones seguras* (1) o si no considera esa posibilidad (0).
Los valores **por defecto** para estos parámetros se pueden establecer en la configuración del fichero **php.ini**. Si visualizamos info.php podremos ver –en las directivas **session.cookie_xxx** nuestros valor por defecto– y observaremos que **session.cookie_lifetime** tiene valor **0**, razón por la cual si –con la opción aceptar cookies activada– ejecutamos cualquiera de los script anteriores y miramos el directorio *Temporal Internet Files*, no encontraremos ninguna de estas cookies, dado que su plazo de expiración es cero.

```
<?
session_name('mi_sesion');
session_start();
echo session_id(),"<br>";
echo session_name(),"<br>";
# recogemos en la variable $a el array con los datos de la sesión
$a=session_get_cookie_params();
foreach($a as $c=>$v){
        echo $c,"--->",$v,"<br>";
}
?>
<A Href="ejemplo122.php">Volver a llamar esta página</A>
```

Ejemplo 122.php

```
<?
session_name('mi_sesion');
session_start();
echo session_id(),"<br>";
echo session_name(),"<br>";
# recogemos en la variable $a el array con los datos de la sesión
$a=session_get_cookie_params();
foreach($a as $c=>$v){
```

sesiones es a través de **cookies**, pero como quiera que el usuario tiene la posibilidad de activar la opción *no aceptar cookies* y eso es algo que no podemos prever, **PHP** dispone una opción alternativa que permite la *propagación* a través de la **URL**.

Caso de que el cliente tenga activada la opción aceptar cookies

Si queremos que las sesiones se propaguen *únicamente* en el caso de que esté activada la opción *aceptar cookies* en el navegador bastará con *hacer la llamada* a la nueva página siguiendo el método tradicional, es decir:

```
<A href="pagxx.php">
```

y que esa nueva página contenga **sin que lo preceda ninguna línea en blanco** el *script* siguiente:

```
<?
session_cache_limiter();
session_name('nombre');
session_start();
.....
?>
```

donde *session_cache_limiter* no es un elemento imprescindible y podremos incluirlo o no, de acuerdo con nuestro interés.

Con **session_name** ocurre algo similar. Si la *sesión* fue iniciada con un nombre distinto al que le asigna por defecto PHP, debemos escribir en el *script* anterior la función **session_name**, y además, debe contener el *mismo nombre* que le habíamos asignado en la página de procedencia.

Si no se asigna ningún nombre tomará **PHPSESSID** –nombre por defecto– en todas las páginas y no será necesaria la instrucción **session_name**.

Caso de cookies desahabilitadas

Para garantizar la propagación de las sesiones –aún cuando esté activada la opción *no aceptar cookies*– tendremos que *pasar* el *identificador de sesión* junto con las llamadas a las páginas siguientes. Para ello se requiere la siguiente sintaxis:

```
<A href="pagxx.php?
<? echo session_name(). "="
.session_id()?>
```

Con esta sintaxis, después de escribir el **?** (recordemos que es la forma de indicar que añadimos variables y valores para que sean transferidos junto con la petición de la página) estaremos pidiendo a PHP que escriba como nombre de variable el nombre de la sesión y como valor de esa variable, después de insertar el signo igual, el identificador de sesión.

Es evidente que la utilización de esta opción nos permite asegurar que las sesiones y sus variables podrán ser usadas sin riesgos de que una configuración inadecuada por parte del cliente produzca resultados imprevisibles.

Ejercicio nº 36

Desarrolla un formulario en el que el usuario pueda escribir su nombre y elegir un color de fondo. Al enviar este formulario los valores de ambos campos se

```
        echo $c,"--->",$v,"<br>";
}
?>
<A Href="ejemplo123.php?<?=$SID ?>">Volver a llamar está página</A>
```

Ejemplo 123.php

```
<?
session_name('mi_sesion');
session_start();
echo session_id(),"<br>";
echo session_name(),"<br>";
# recogemos en la variable $a el array con los datos de la sesión
$a=session_get_cookie_params();
foreach($a as $c=>$v){
        echo $c,"--->",$v,"<br>";
}
?>
<A Href="ejemplo124.php?<?echo session_name()."=".session_id()?>">
                Volver a llamar esta página</A>
```

Ejemplo 124.php

Las configuraciones por defecto de **session.cookie** pueden cambiarse sin necesidad de modificar el fichero php.ini.

Mediante la función **session_set_cookie_params(***duración ,'path','dominio',segura***)** se pueden configurar esos parámetros de forma temporal–**unicamente para la página en la que está insertado**– y que como en los casos anteriores tiene tres posibilidades de *script*.

En los ejemplos siguientes, se *escribirá* -siempre que esté configurada la opción **aceptar cookies** en el navegador del cliente- una *cookie* que tendrá una caducidad de *10 minutos* (el valor **10** de *session_set_cookie_params*).

Esa cookie se guardará en el mismo directorio (/) donde se guardan las páginas web visitadas (el famoso *C:\WINDOWS\Temporary Internet Files*, en la configuración por defecto de IE), pero..., eso solo ocurrirá si -tal como ves en el ejemplo- pones en el parámetro **dominio** el **nombre real del dominio donde está alojada la web**.

Si el nombre de dominio no coincide con que alberga la página –razonable criterio de seguridad– **no se guardará** la *cookie*.

```
<?
session_set_cookie_params (10,"/","localhost", 0);
session_name('mi_sesion');
session_start();
echo session_id(),"<br>";
echo session_name(),"<br>";
$a=session_get_cookie_params();
foreach($a as $c=>$v){
        echo $c,"--->",$v,"<br>";
}
?>
<A Href="ejemplo125.php">Volver a llamar esta página</A>
```

Ejemplo 125.php

```
<?
session_set_cookie_params (10,"/","localhost", 0);
session_name('mi_sesion');
session_start();
echo session_id(),"<br>";
echo session_name(),"<br>";
$a=session_get_cookie_params();
foreach($a as $c=>$v){
        echo $c,"--->",$v,"<br>";
}
?>
<A Href="ejemplo126.php?<?=$SID ?>">Volver a llamar esta página</A>
```

Ejemplo 126.php

```
<?
session_set_cookie_params (10,"/","localhost", 0);
session_name('mi_sesion');
session_start();
echo session_id(),"<br>";
echo session_name(),"<br>";
$a=session_get_cookie_params();
foreach($a as $c=>$v){
        echo $c,"--->",$v,"<br>";
}
?>
<A Href="ejemplo127.php?<?echo session_name()."=".session_id()?>">
                Volver a llamar esta página</A>
```

registrarán en variables de sesión y se visualizará una nueva página que tendrá el color de fondo elegido y que presentará el nombre de usuario. Además, esta página, contendrá un enlace a una segunda página a la que deberán propagarse los valores anteriores.

Ejemplo 127.php

¡Cuidado!

Antes de empezar a desarrollar el ejercicio que te proponemos al margen, te sugerimos que leas los contenidos que hemos incluido en la página siguiente.